Introduction to Artificial Intelligence

丁尧相
浙江大学

Fall & Winter 2022
Week 14

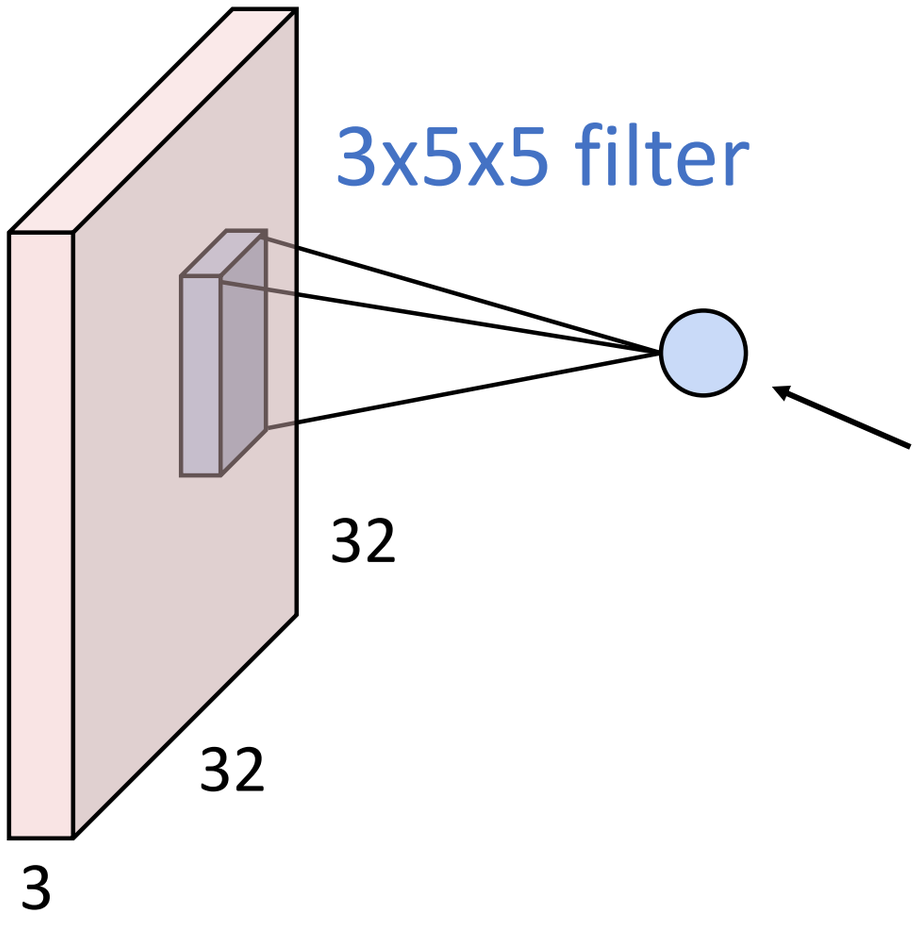
Announcements

- Problem set 3 is due today.
- Problem set 4 (last homework) will be released this week.
- Lab project 3 (last lab project) will be released this week.
- The final exam is scheduled on Jan. 7. The deadlines of lab projects will extended to Jan. 7. (recommended to submit before Jan. 1).

Machine Learning: V

- NN to process sequential data
  - Recurrent NN
- Generative NN models
  - Autoencoders
  - GANs
- Take-home messages

Convolution Layer



1 number:
the result of taking a dot product between the filter and a small 3x5x5 chunk of the image
(i.e. 3*5*5 = 75-dimensional dot product + bias)

$$w^T x + b$$

Convolution Layer

For simplicity, we only show $3 \times 2 \times 2$ filter

x

0.4	0.2
-0.3	-0.5

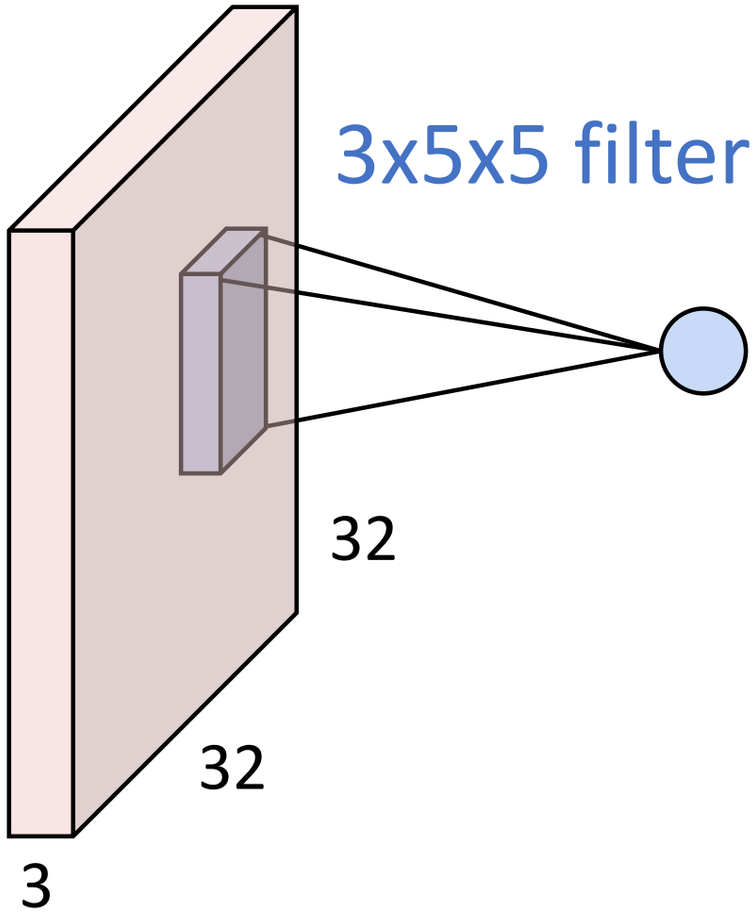
-0.3	-0.2
0.5	0.6

w

0.7	-0.2
0.8	-0.3

0.9	-0.5
0.3	0.4

3x32x32 image



-0.4	0.3
-0.8	0.5

0.8	0.7
-0.5	0.6

1 number:
the result of taking a dot product between the filter and a small 3x5x5 chunk of the image
(i.e. 3*5*5 = 75-dimensional dot product + bias)

$$w^T x + b$$

Convolution Layer

For simplicity, we only show $3 \times 2 \times 2$ filter

Dot products are conducted by stretching matrix into vector

x

0.4	0.2
-0.3	-0.5

dot product

w

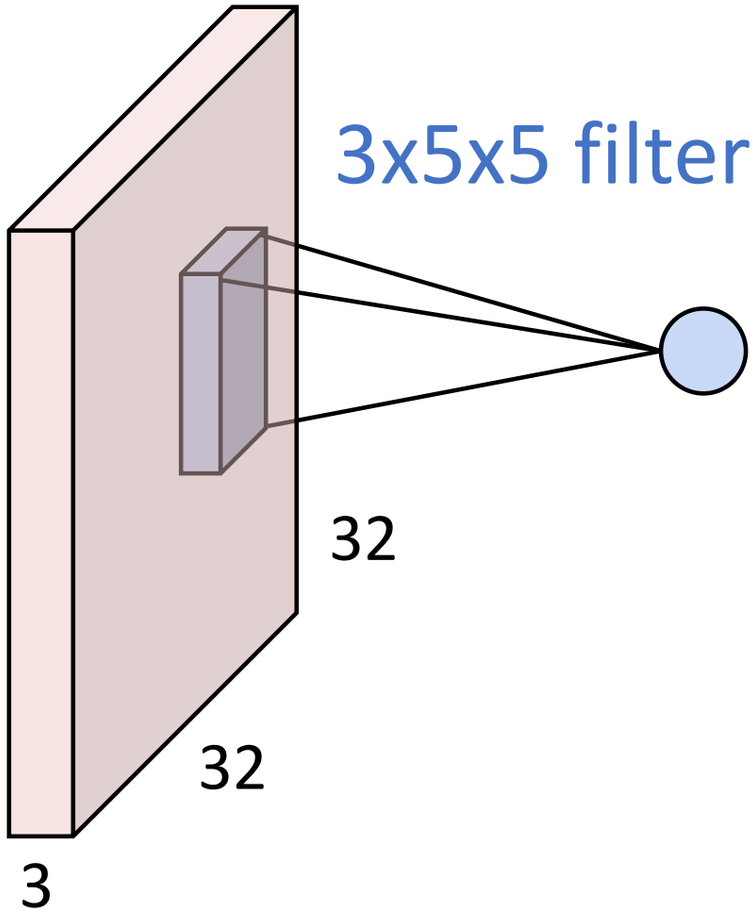
0.7	-0.2
0.8	-0.3

-0.3	-0.2
0.5	0.6

dot product

0.9	-0.5
0.3	0.4

3x32x32 image



-0.4	0.3
-0.8	0.5

dot product

0.8	0.7
-0.5	0.6

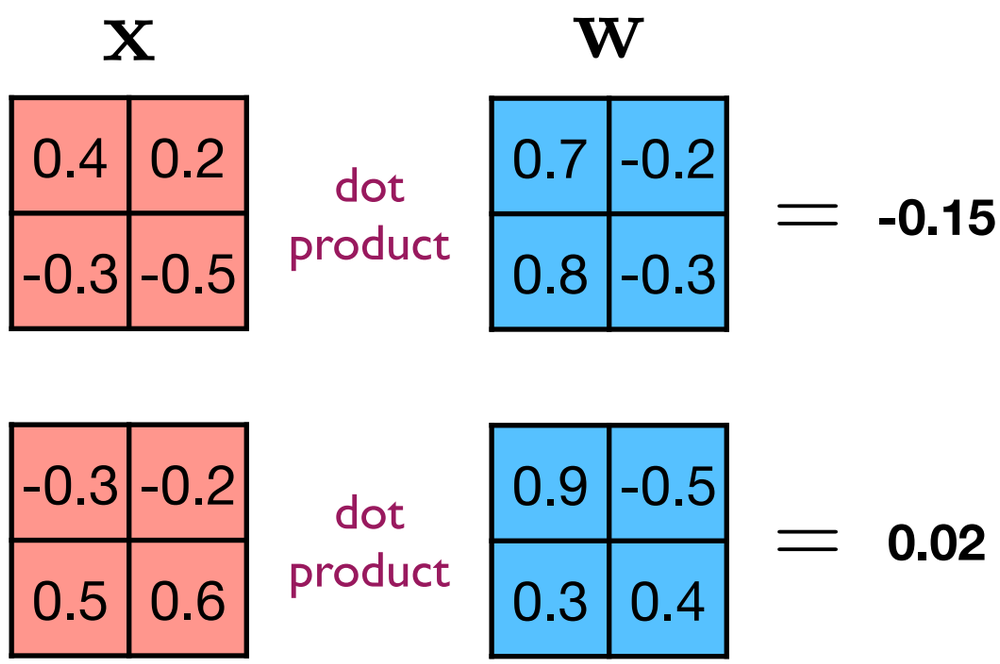
1 number:
the result of taking a dot product between the filter and a small 3x5x5 chunk of the image
(i.e. 3*5*5 = 75-dimensional dot product + bias)

$$w^T x + b$$

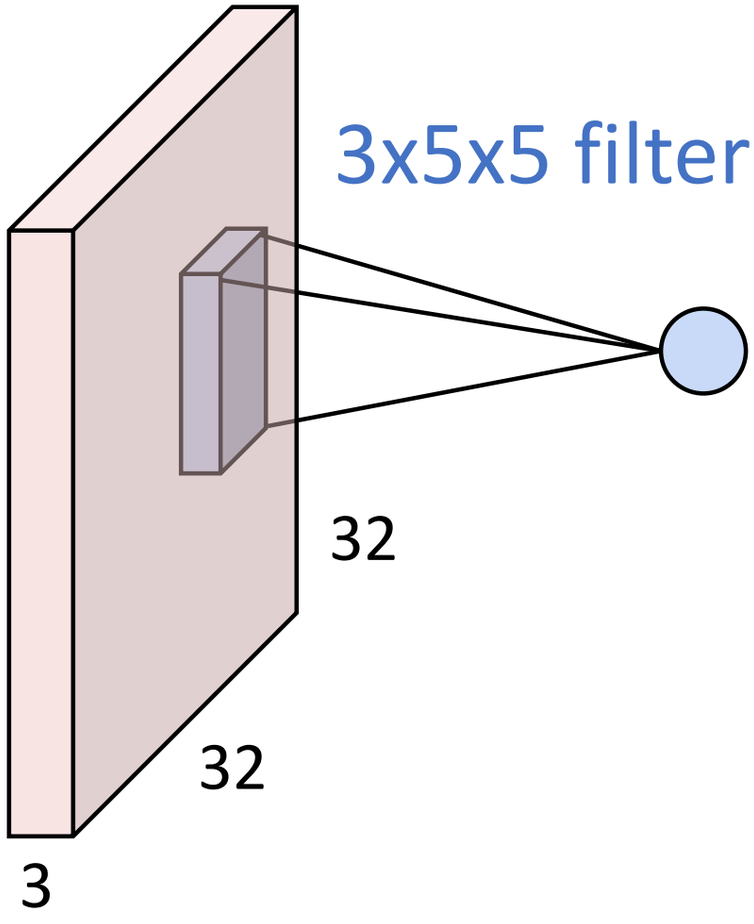
Convolution Layer

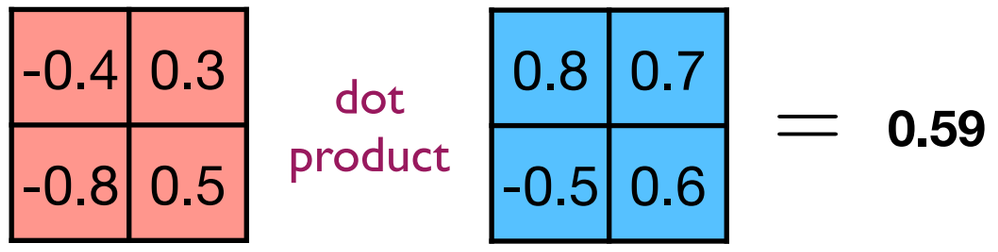
For simplicity, we only show $3 \times 2 \times 2$ filter

Dot products are conducted by stretching matrix into vector



3x32x32 image





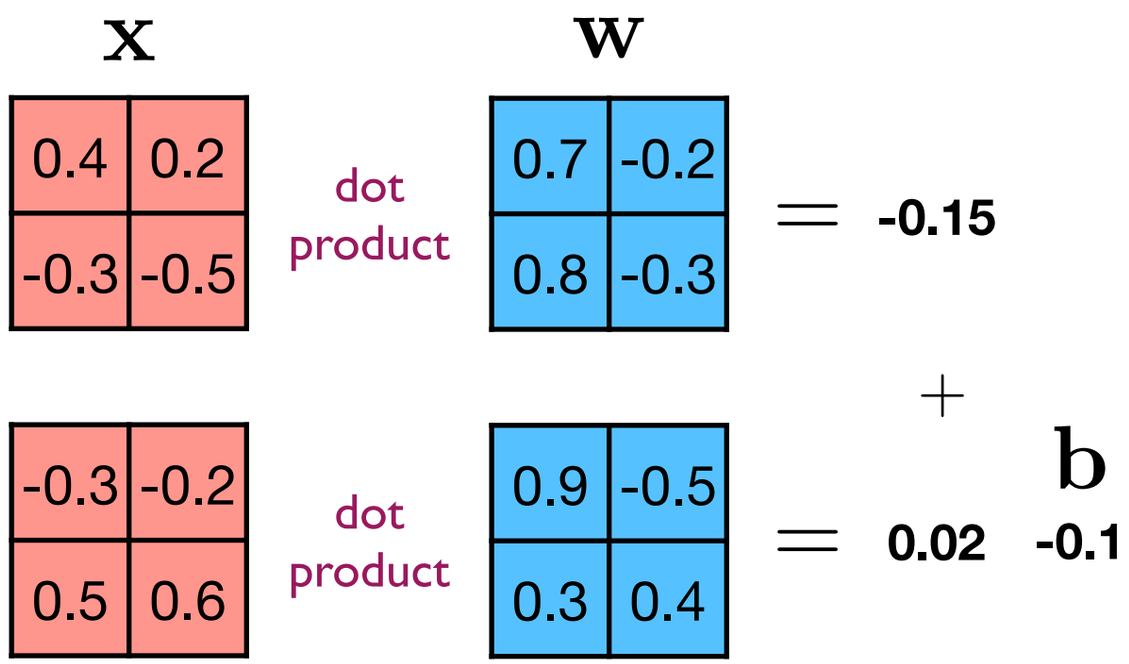
1 number:
the result of taking a dot product between the filter and a small 3x5x5 chunk of the image
(i.e. 3*5*5 = 75-dimensional dot product + bias)

$$w^T x + b$$

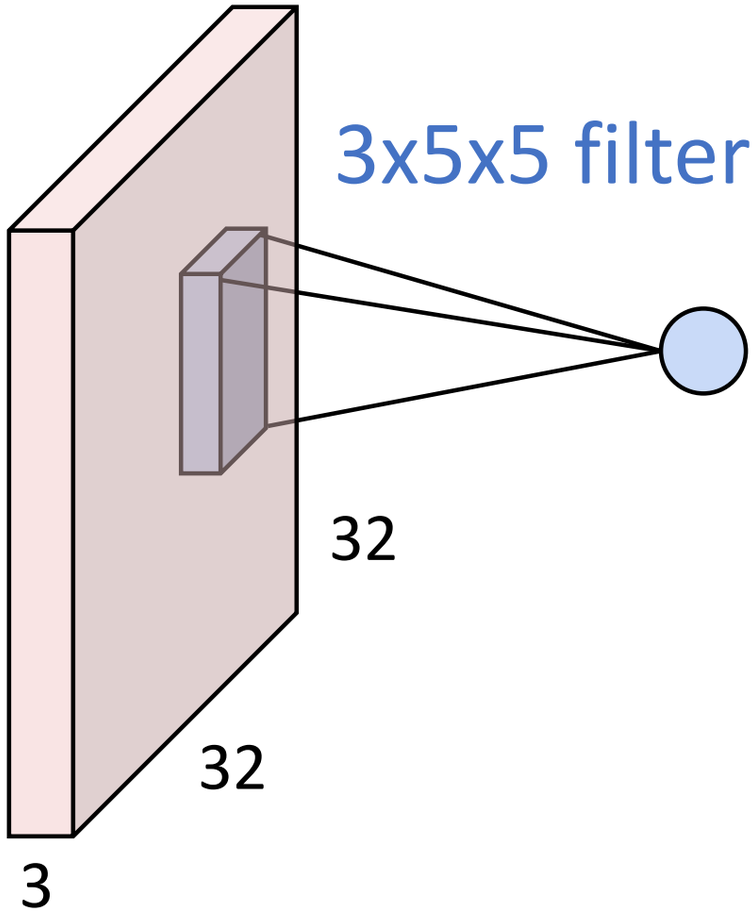
Convolution Layer

For simplicity, we only show $3 \times 2 \times 2$ filter

Dot products are conducted by stretching matrix into vector



3x32x32 image



1 number:
the result of taking a dot product between the filter and a small 3x5x5 chunk of the image
(i.e. 3*5*5 = 75-dimensional dot product + bias)

$$w^T x + b$$

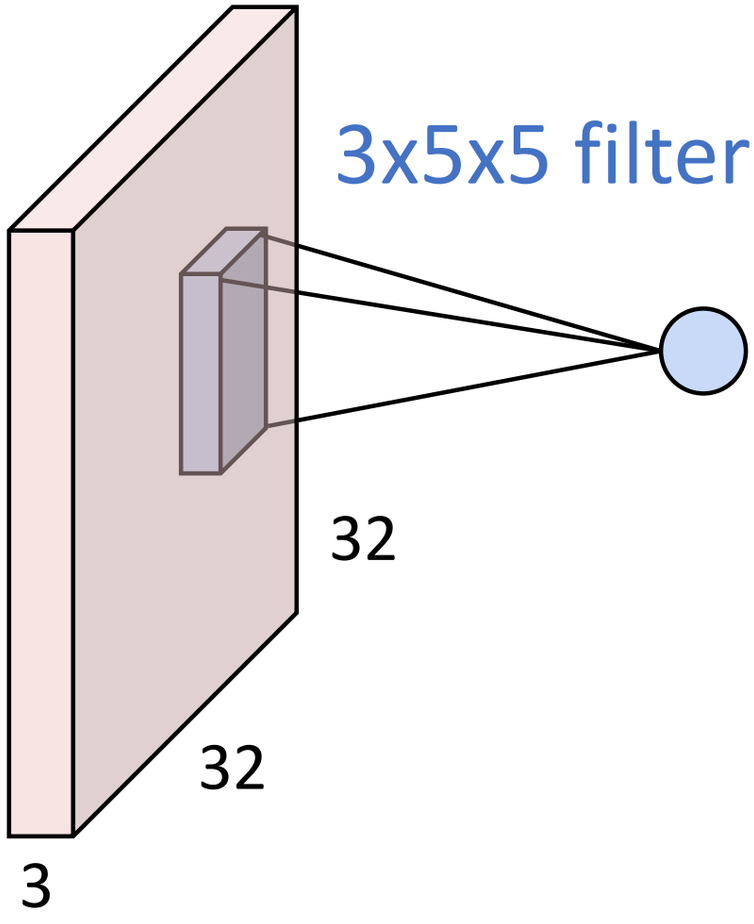
Convolution Layer

For simplicity, we only show $3 \times 2 \times 2$ filter

x		w	
0.4	0.2	0.7	-0.2
-0.3	-0.5	0.8	-0.3

-0.3	-0.2	conv	0.9	-0.5	= 0.36
0.5	0.6		0.3	0.4	

3x32x32 image



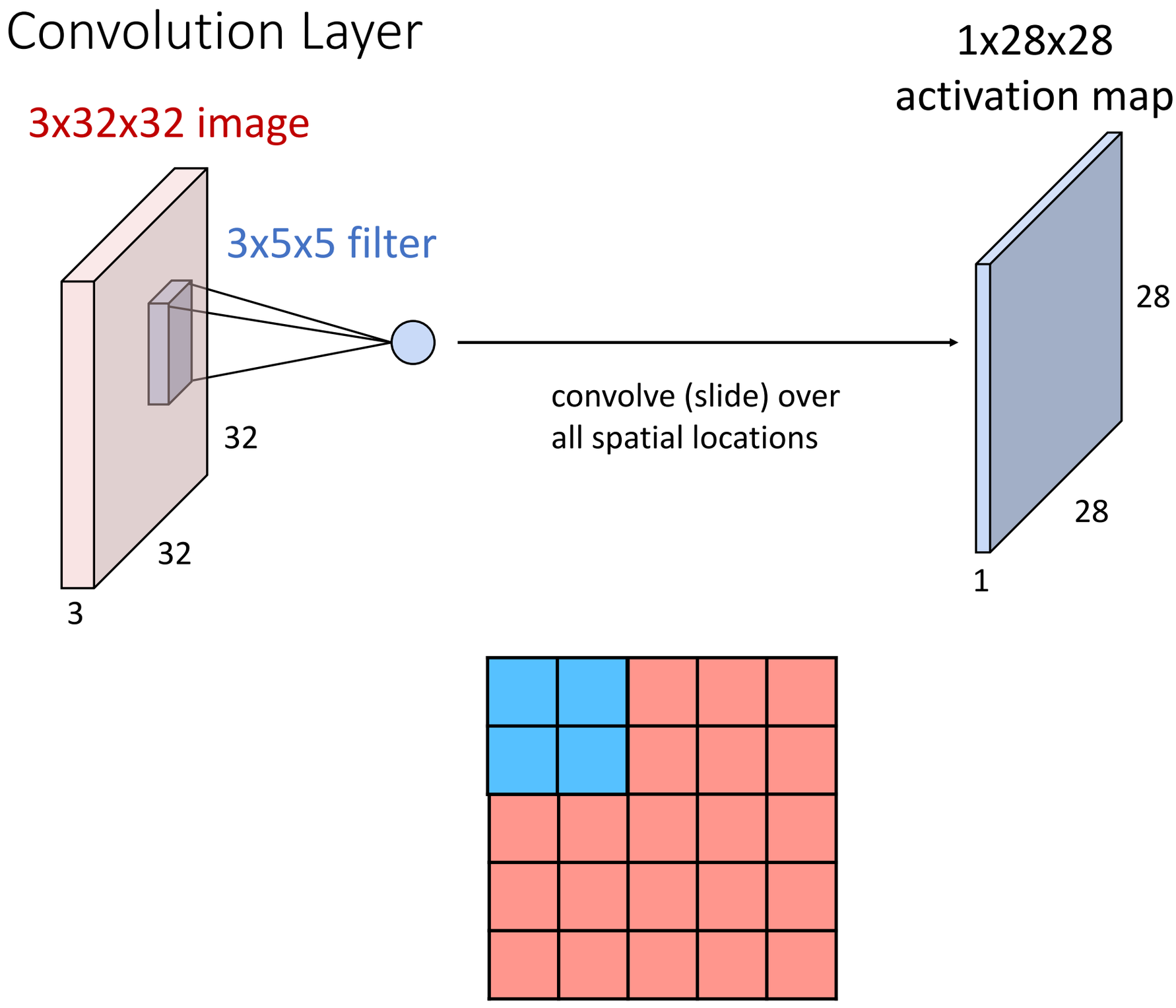
-0.4	0.3
-0.8	0.5

0.8	0.7
-0.5	0.6

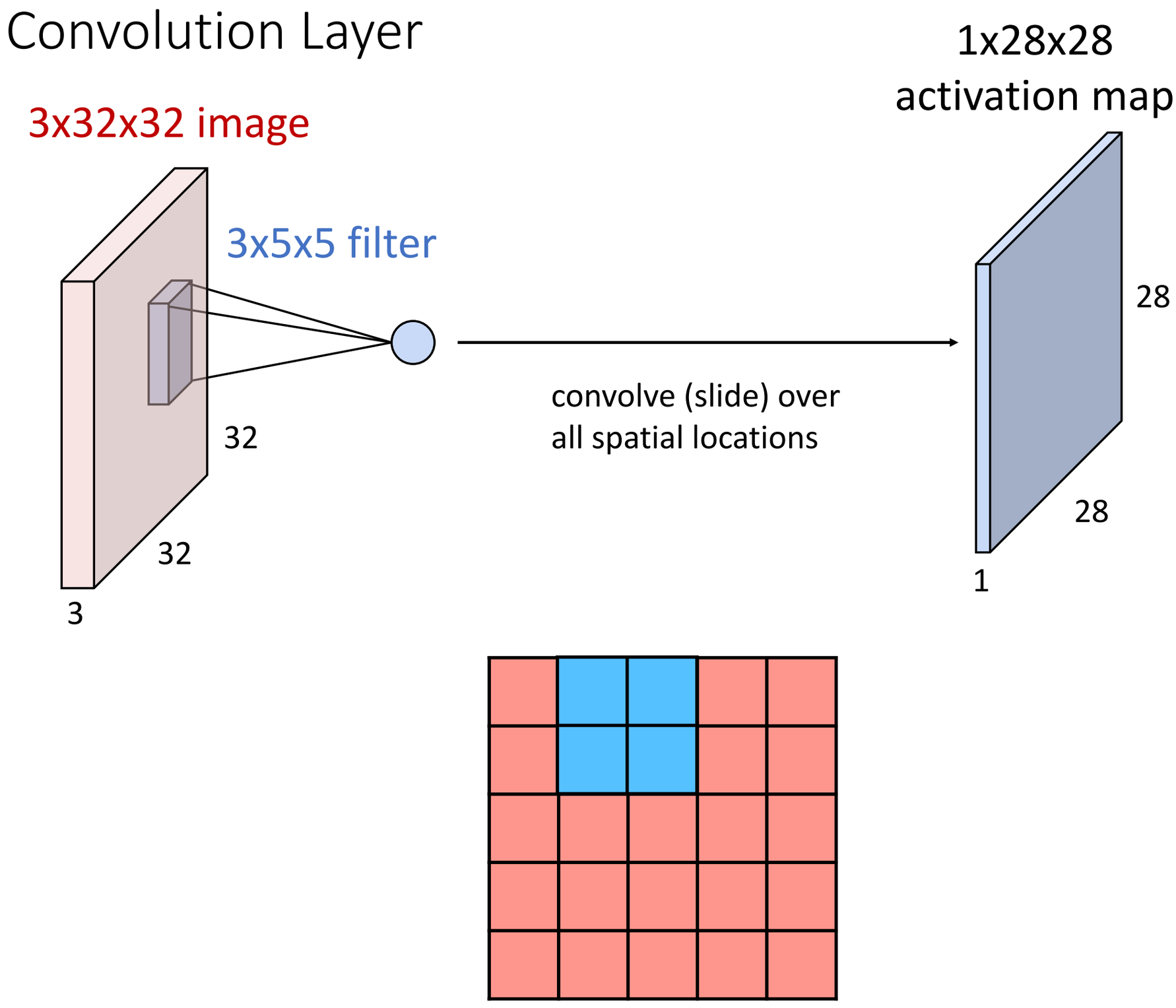
1 number:
the result of taking a dot product between the filter and a small 3x5x5 chunk of the image
(i.e. 3*5*5 = 75-dimensional dot product + bias)

$$w^T x + b$$

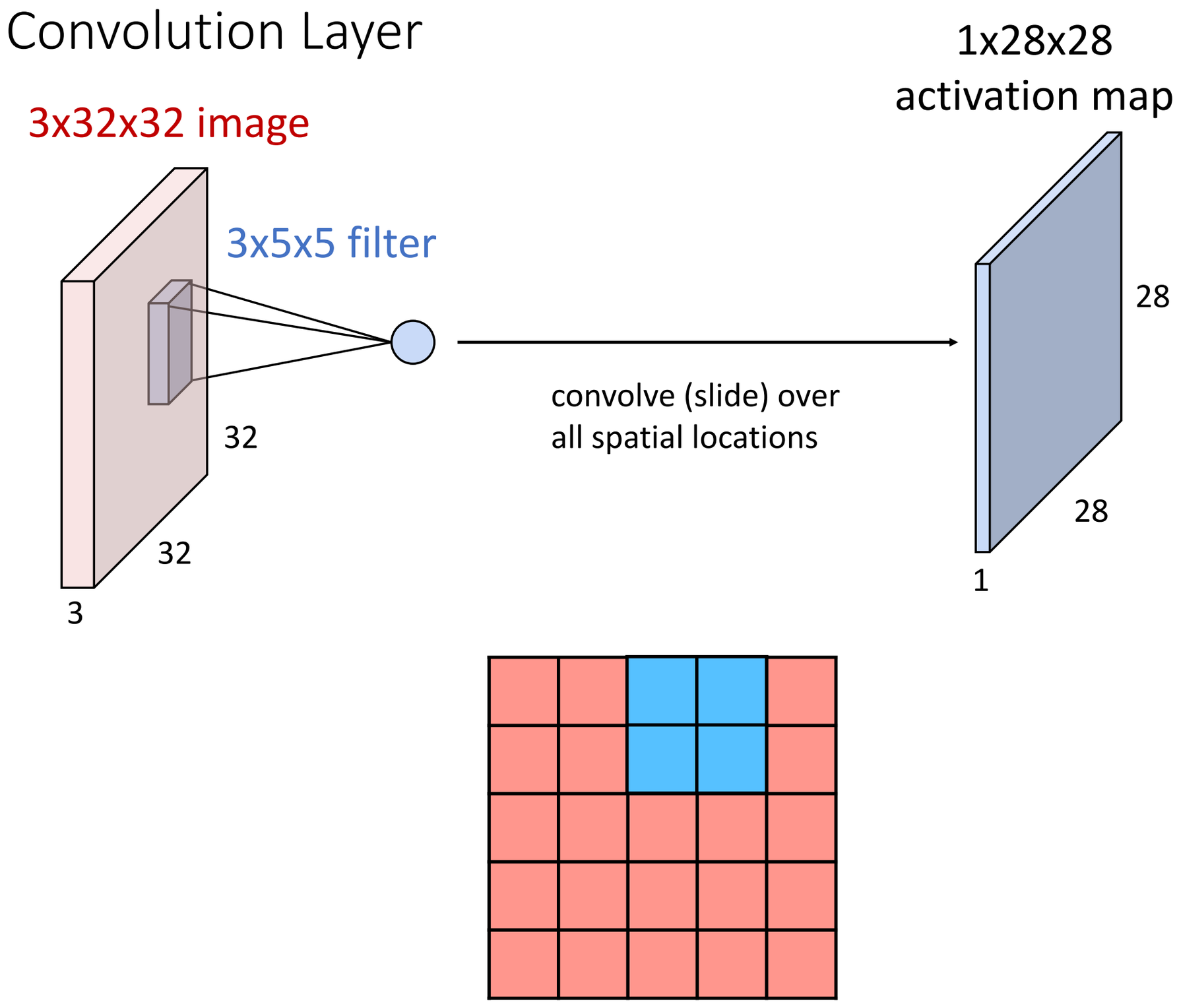
Convolution Layer



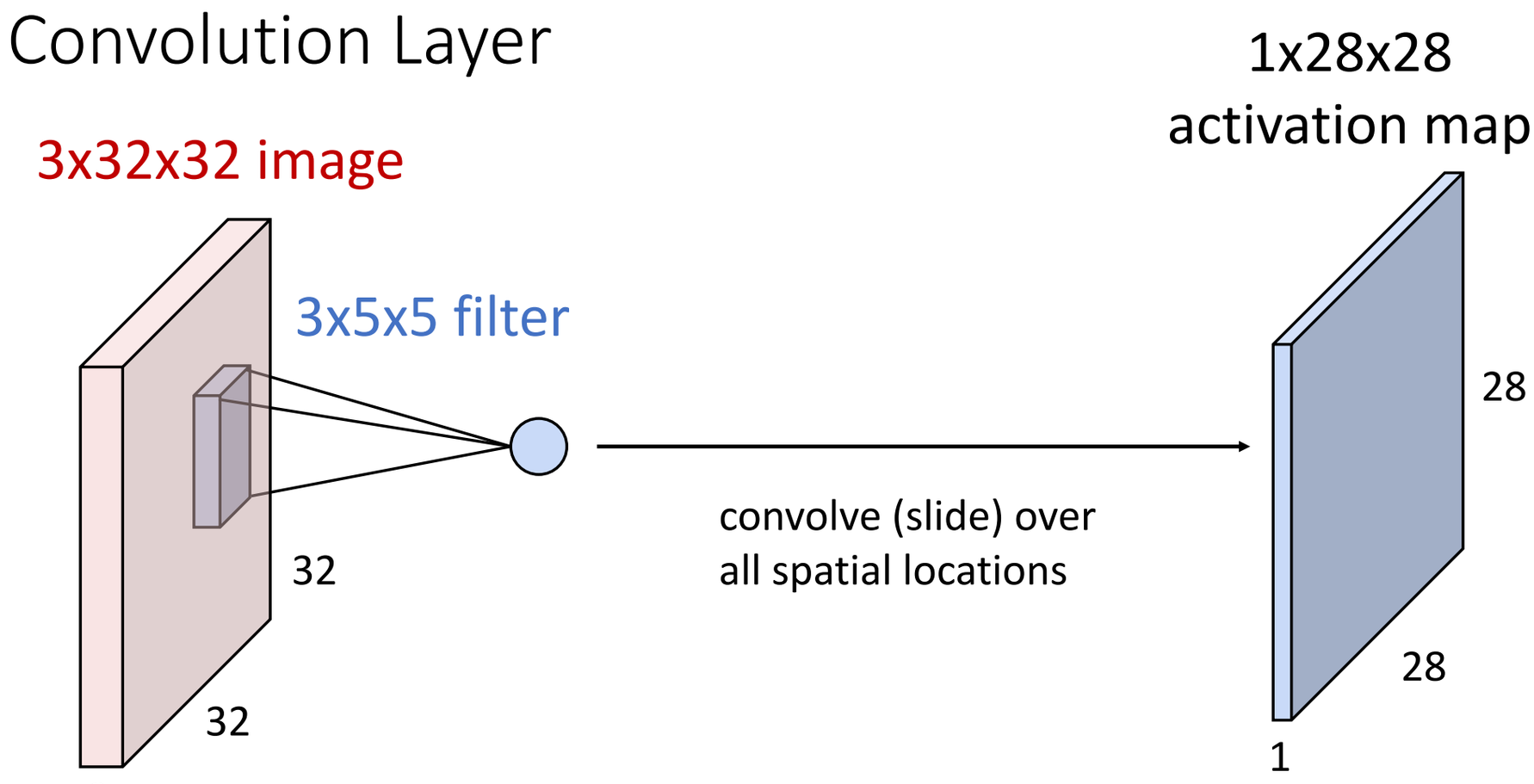
Convolution Layer

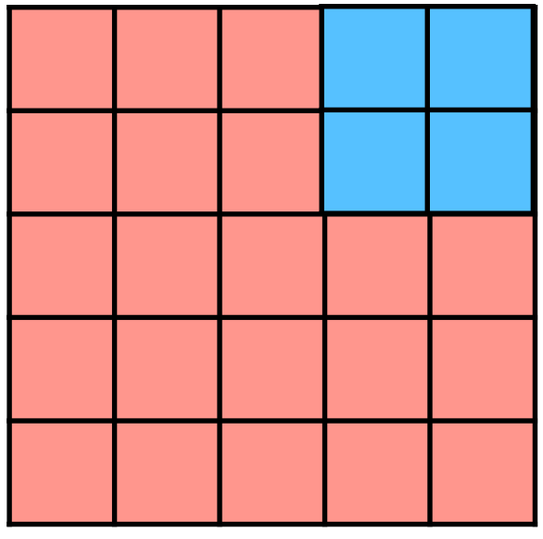


Convolution Layer

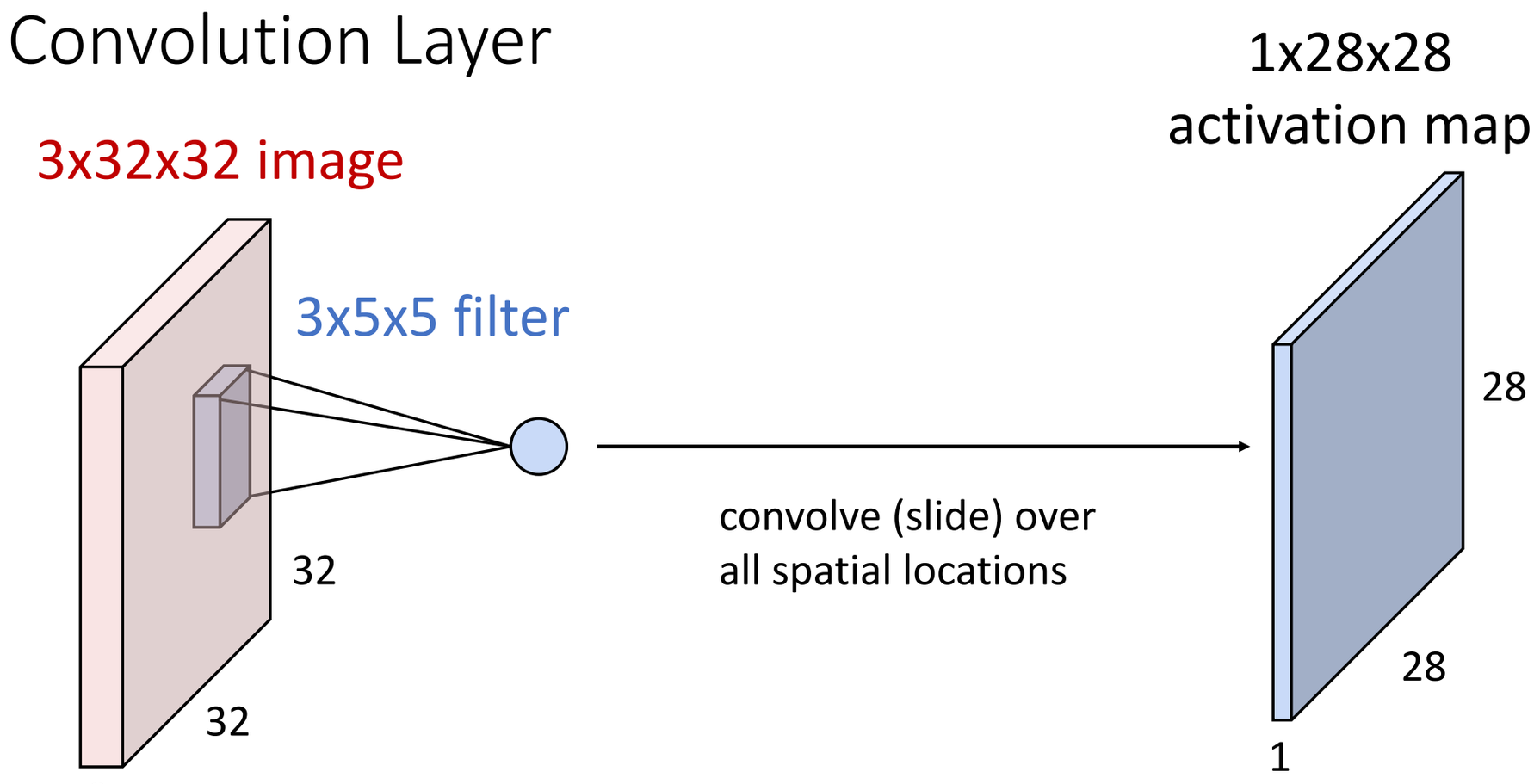


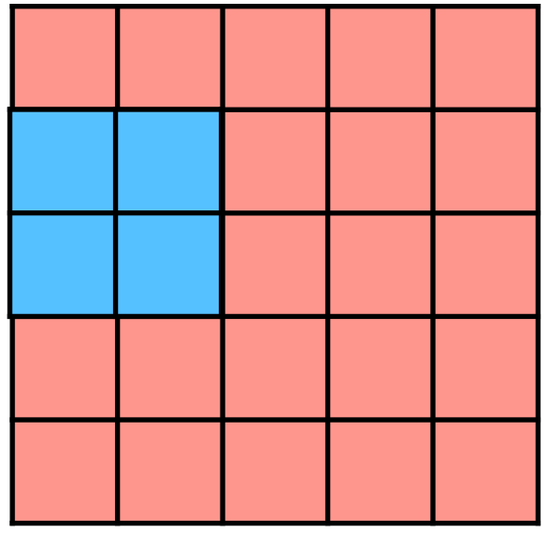
Convolution Layer



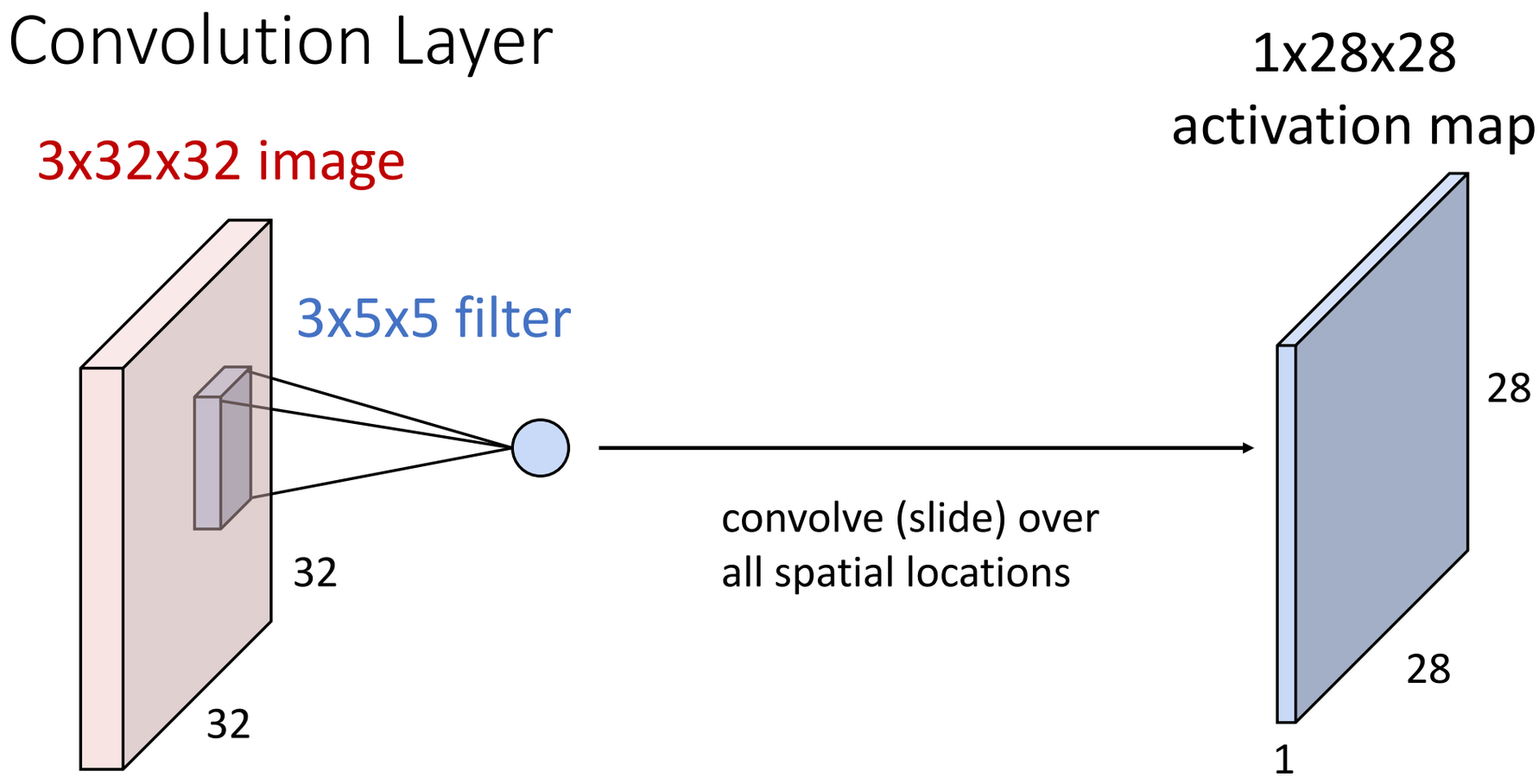


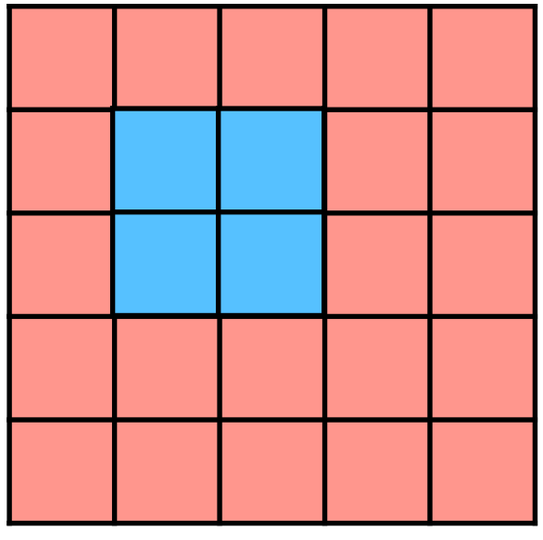
Convolution Layer



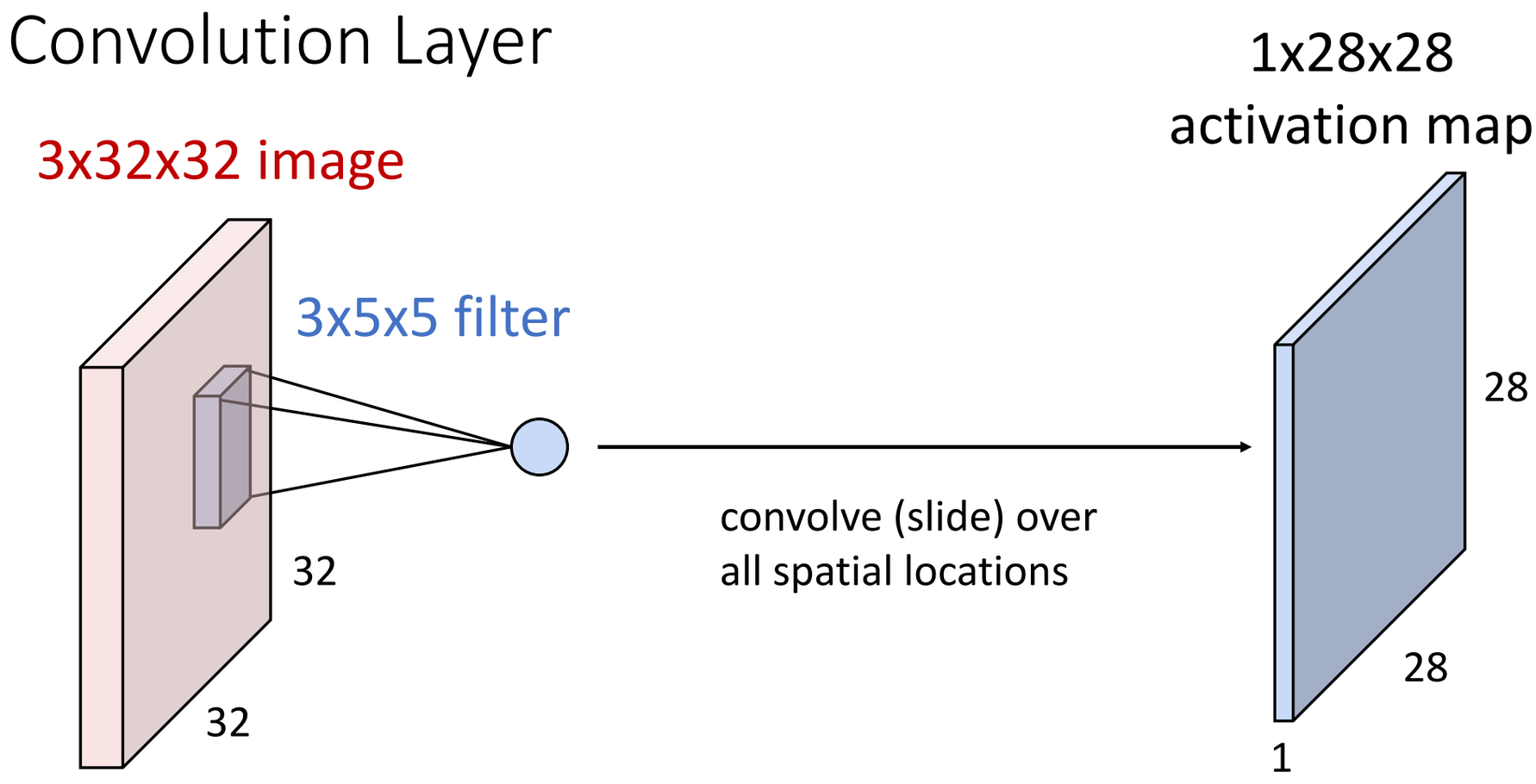


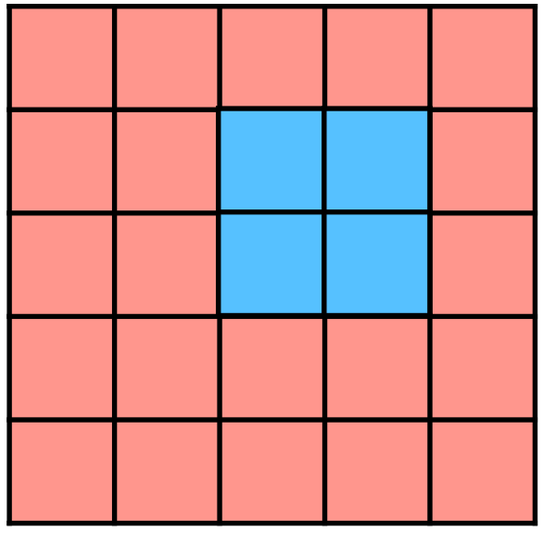
Convolution Layer



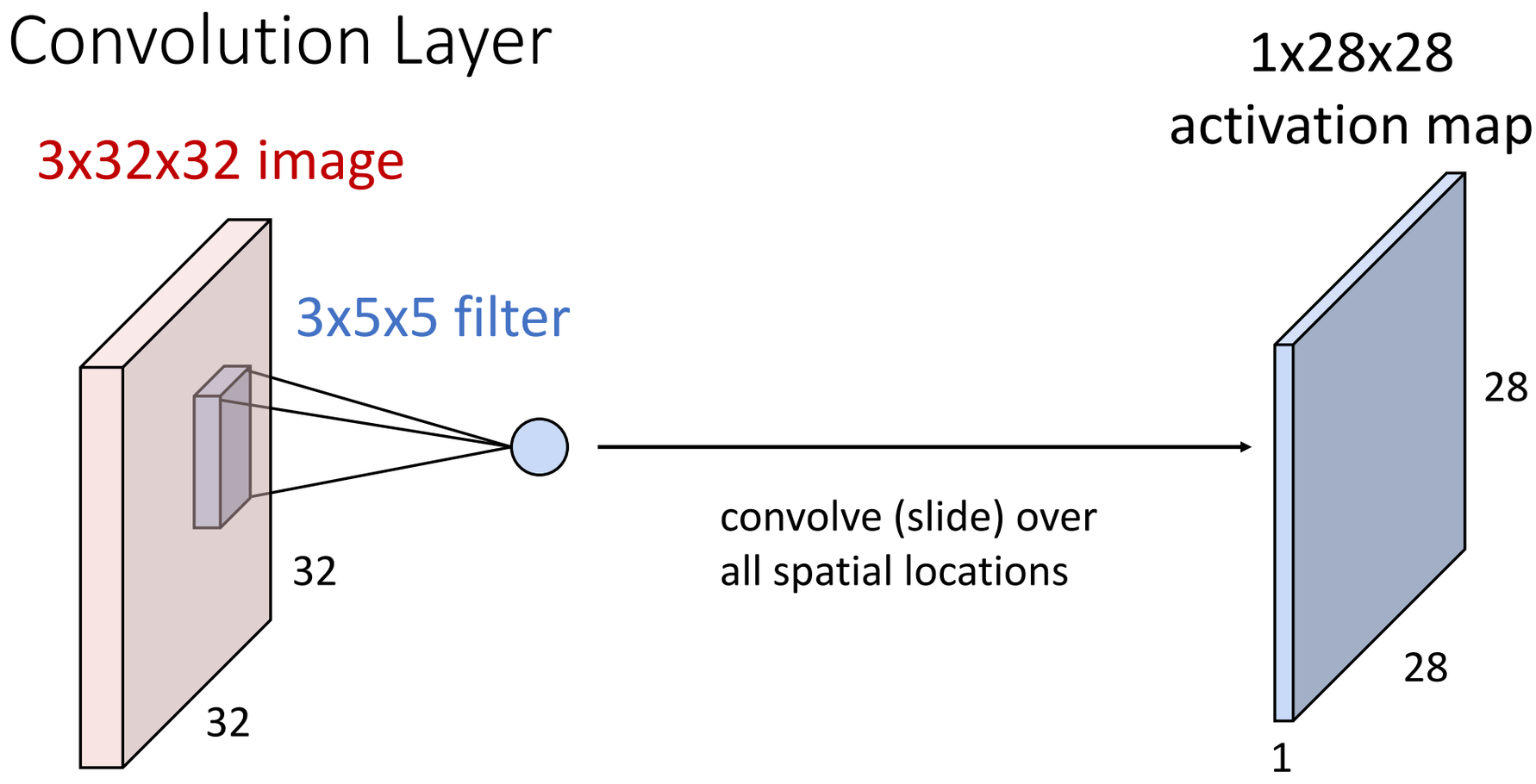


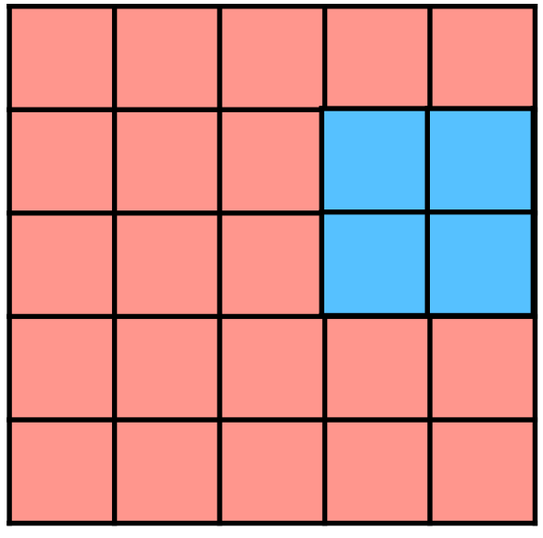
Convolution Layer



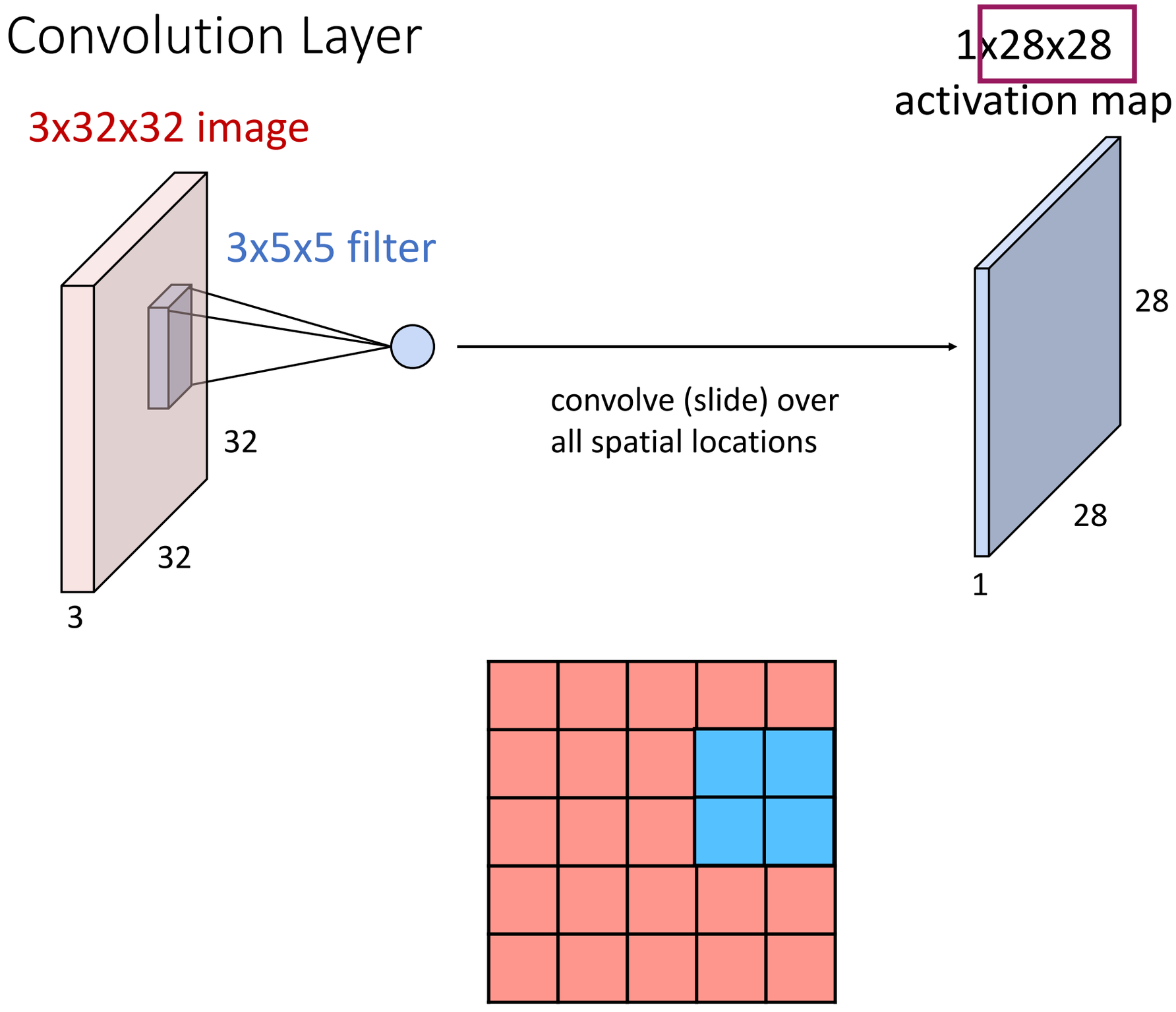


Convolution Layer

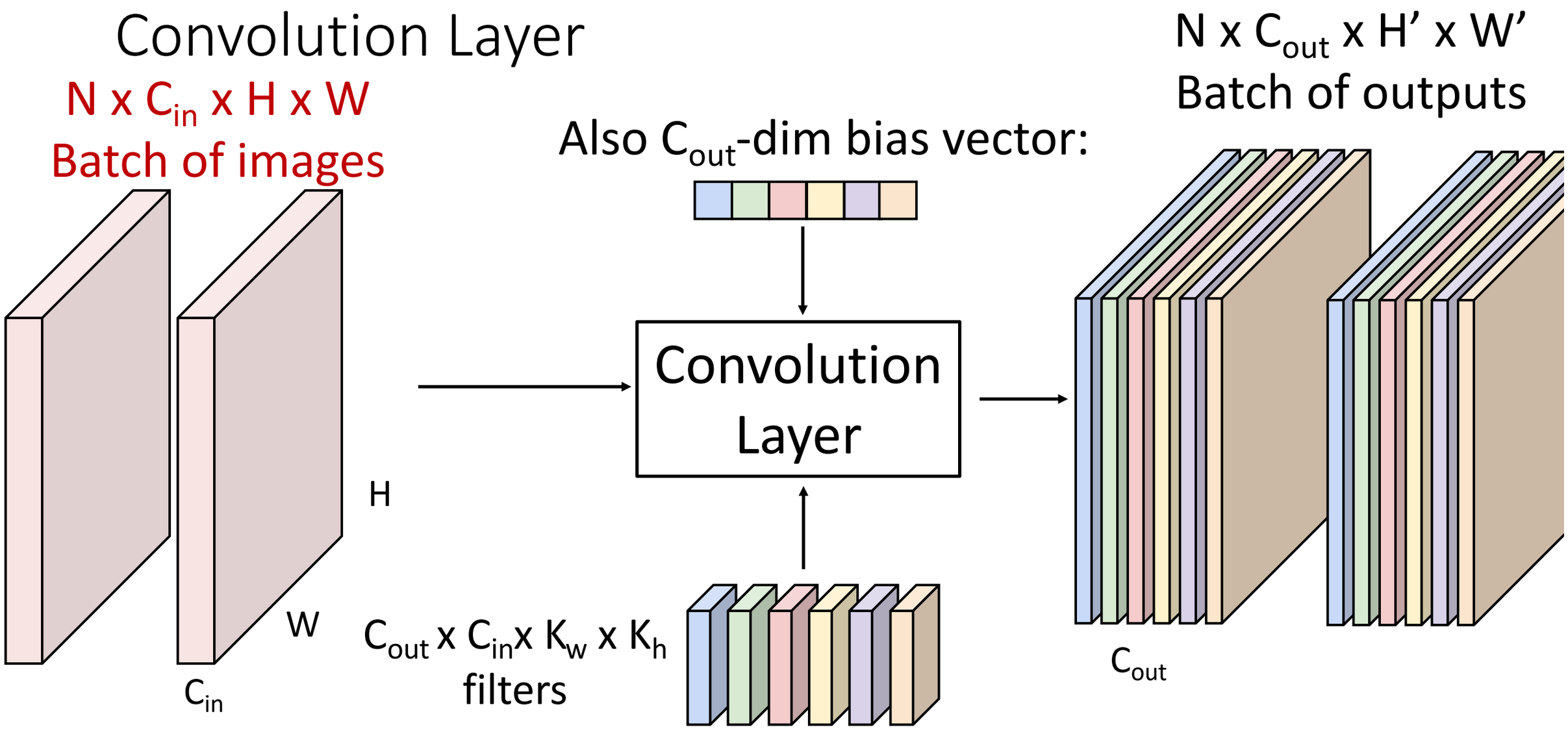




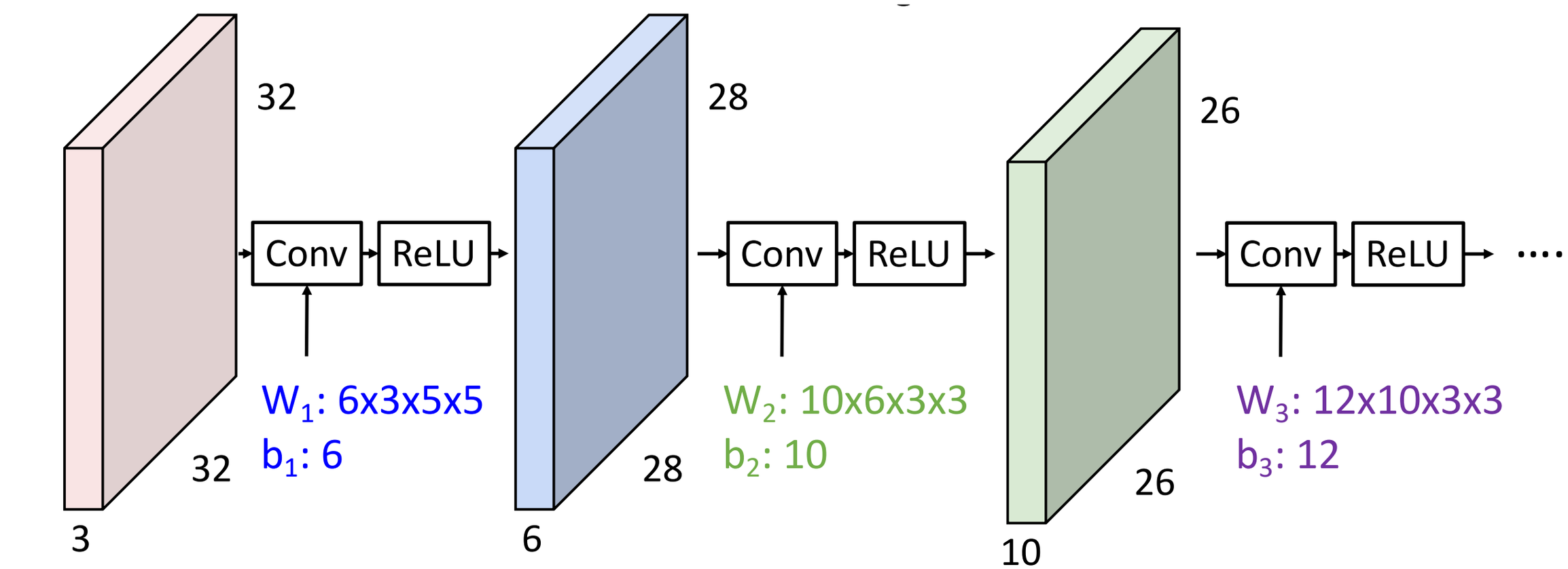
Convolution Layer



Convolution Layer



Convolution Layer



Input:
N x 3 x 32 x 32

First hidden layer:
N x 6 x 28 x 28

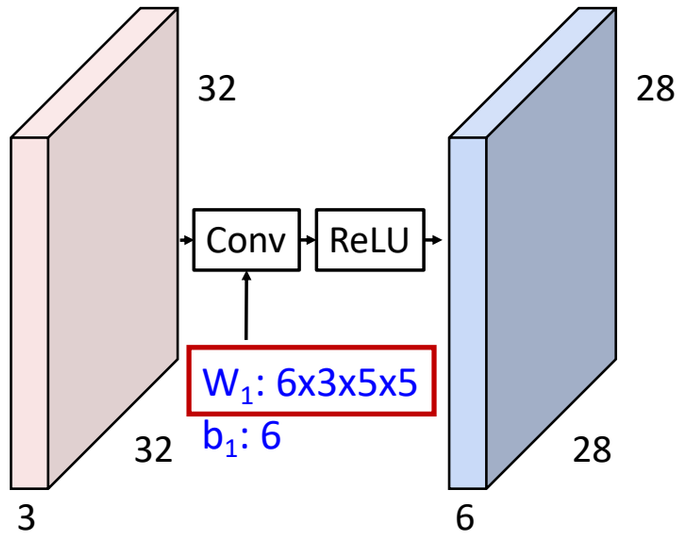
Second hidden layer:
N x 10 x 26 x 26

Convolutions are linear transformations.
Need to add activation functions to obtain non-linearity.

Much fewer parameters than fully-connected layer!

Linear Transformations are Templates

Linear classifier: One template per class

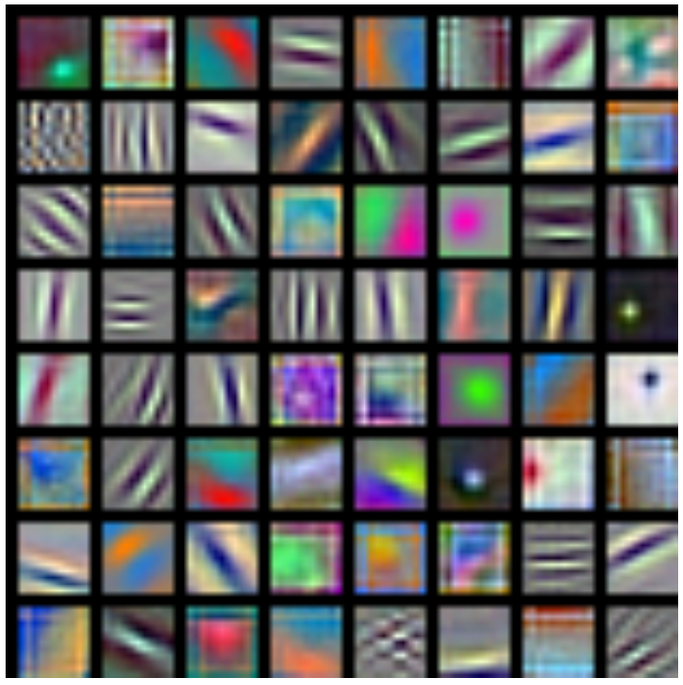


Input:
N x 3 x 32 x 32

First hidden layer:
N x 6 x 28 x 28

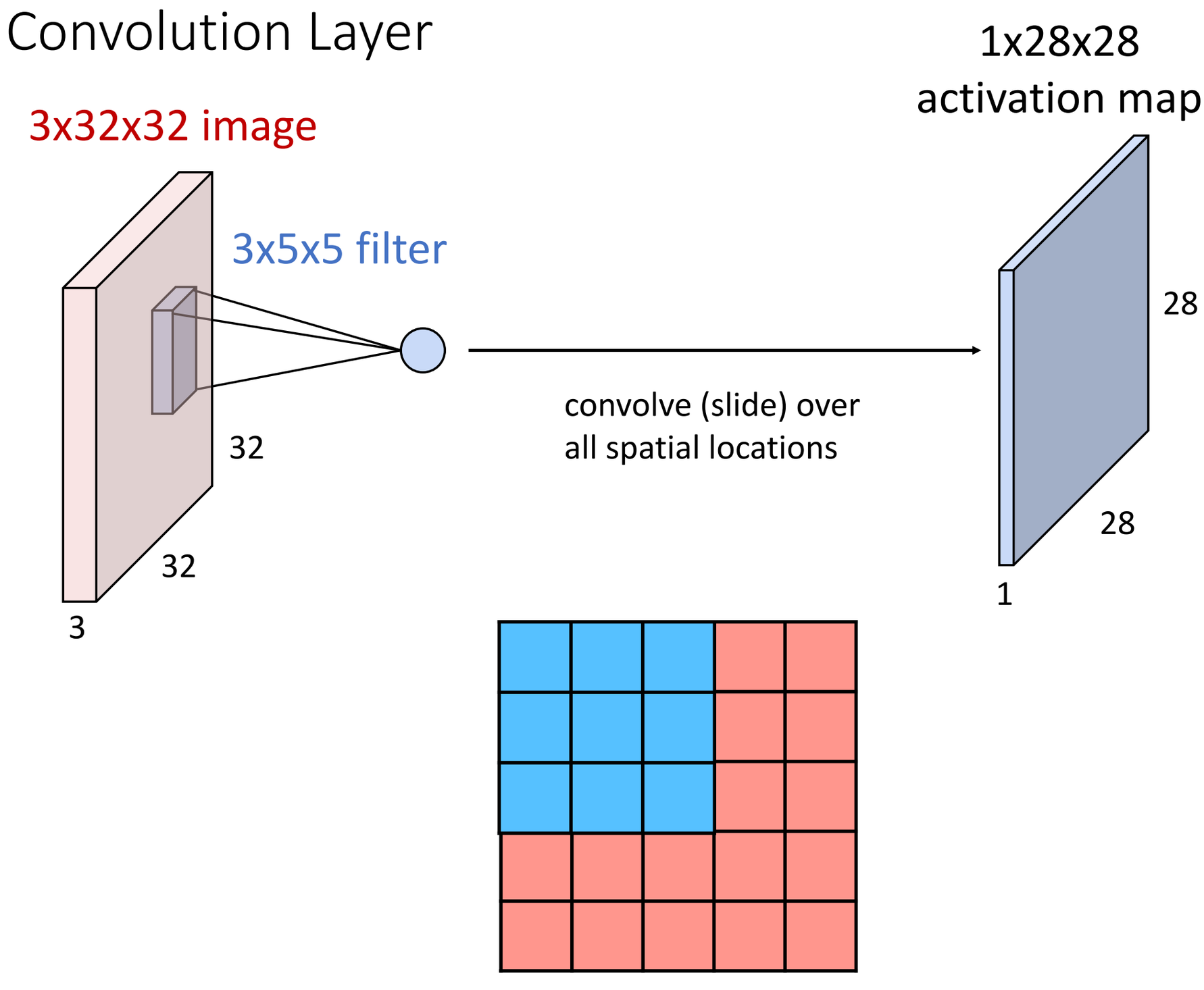
MLP: Bank of whole-image templates

First-layer conv filters: local image templates
(Often learns oriented edges, opposing colors)

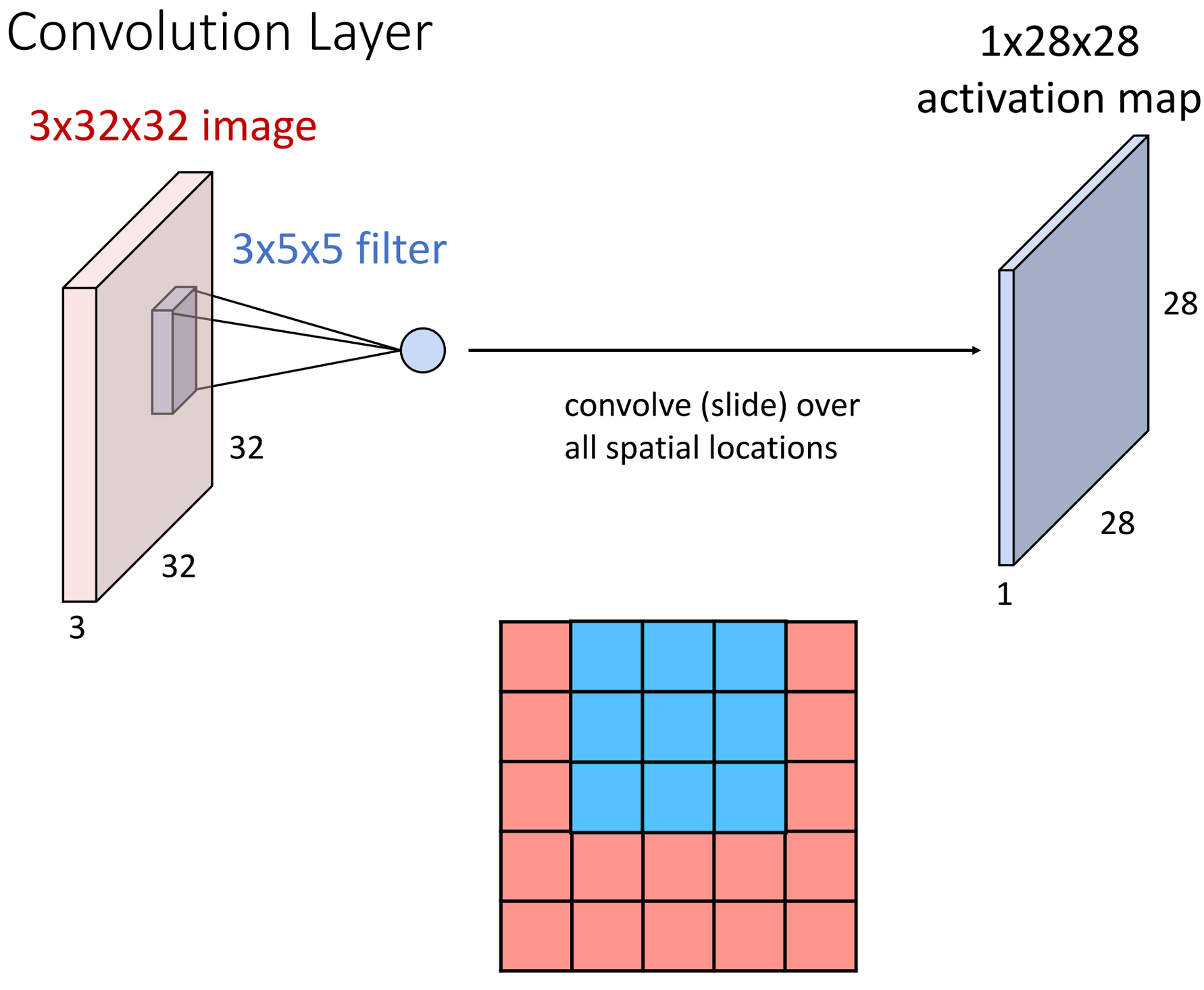


AlexNet: 64 filters, each 3x11x11

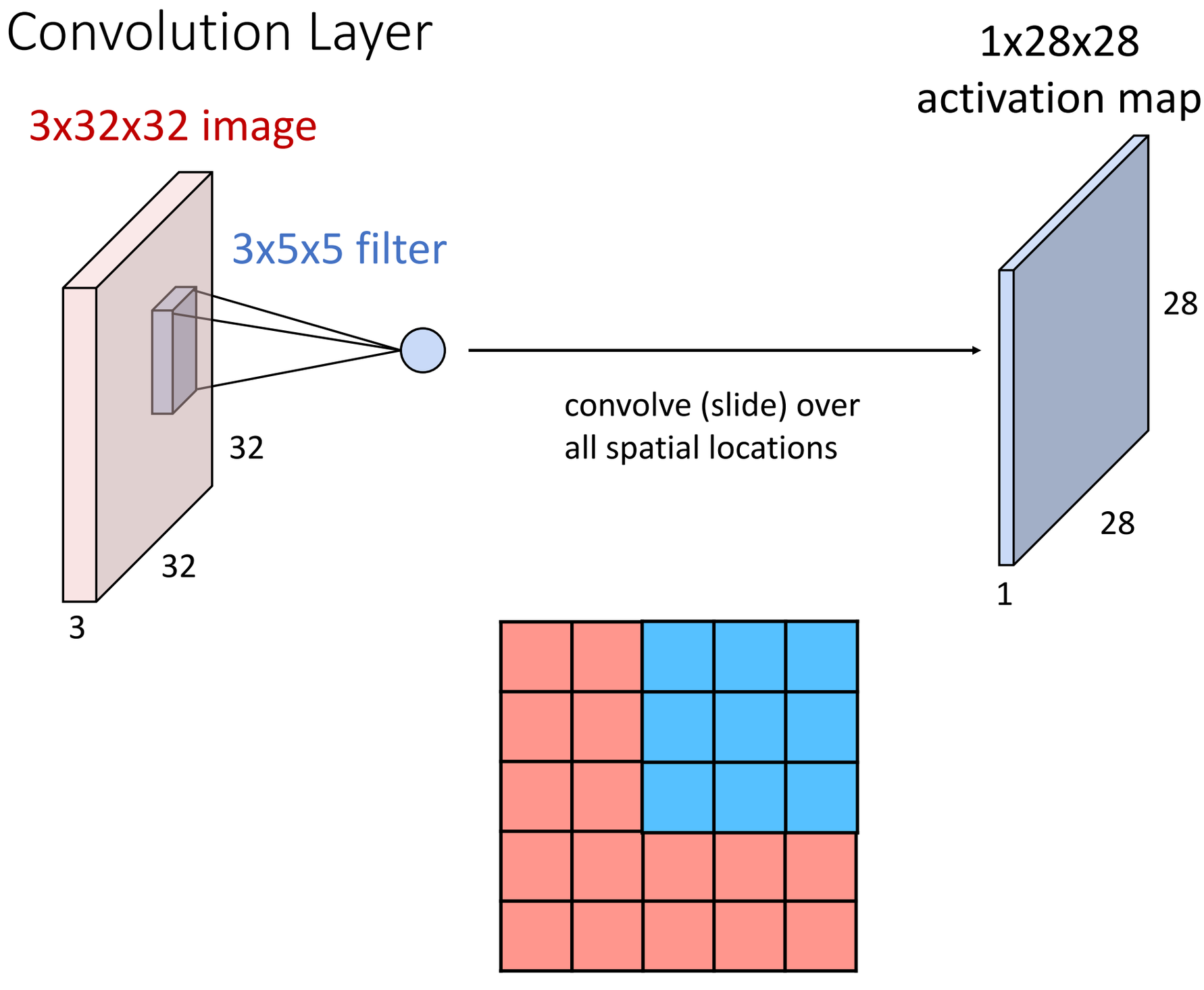
Padding


Convolution Layer
3x32x32 image
3x5x5 filter
32
32
3
convolve (slide) over
all spatial locations
1x28x28
activation map
28
28
1

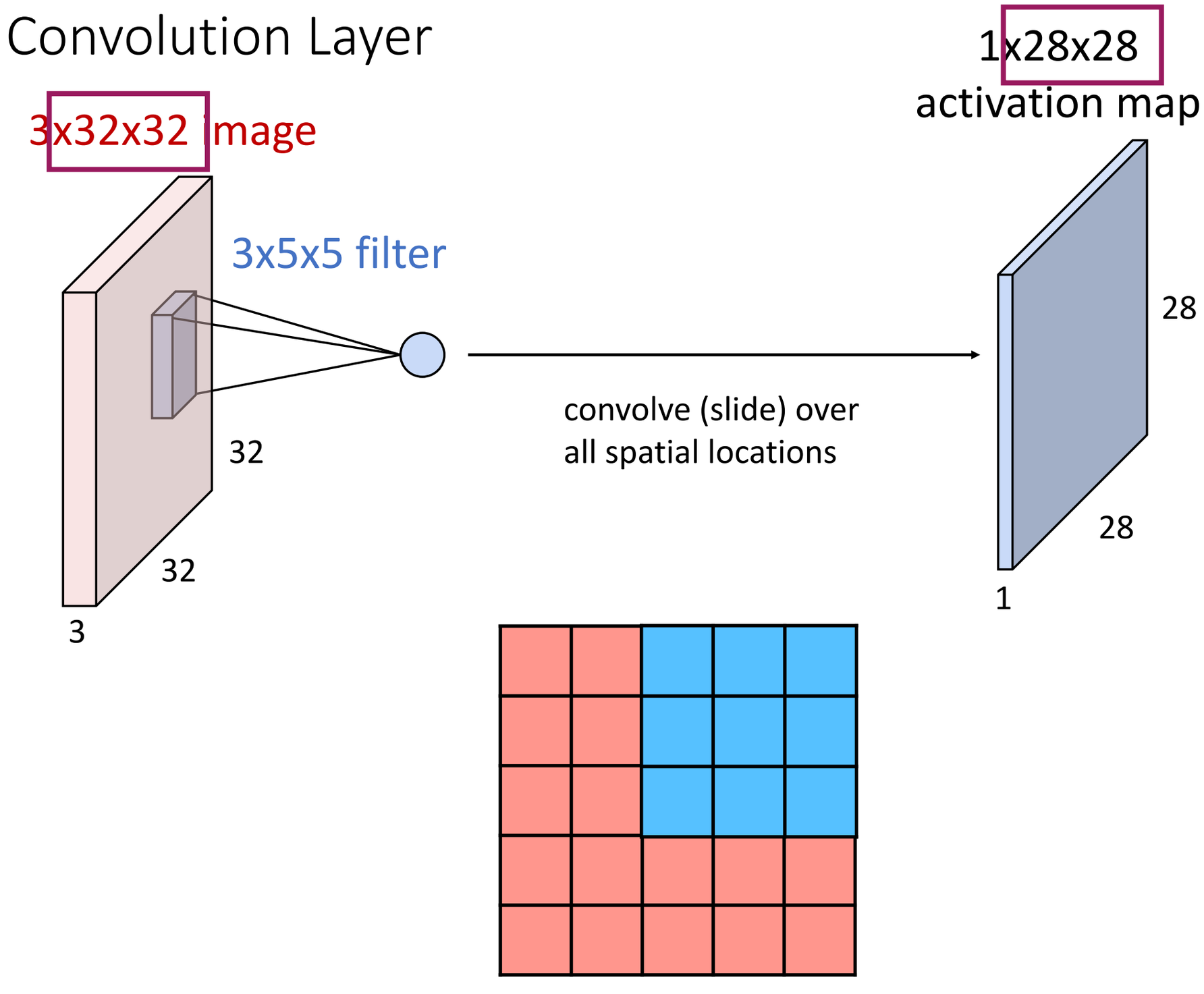
Padding


Convolution Layer
3x32x32 image
3x5x5 filter
32
32
3
convolve (slide) over
all spatial locations
1x28x28
activation map
28
28
1

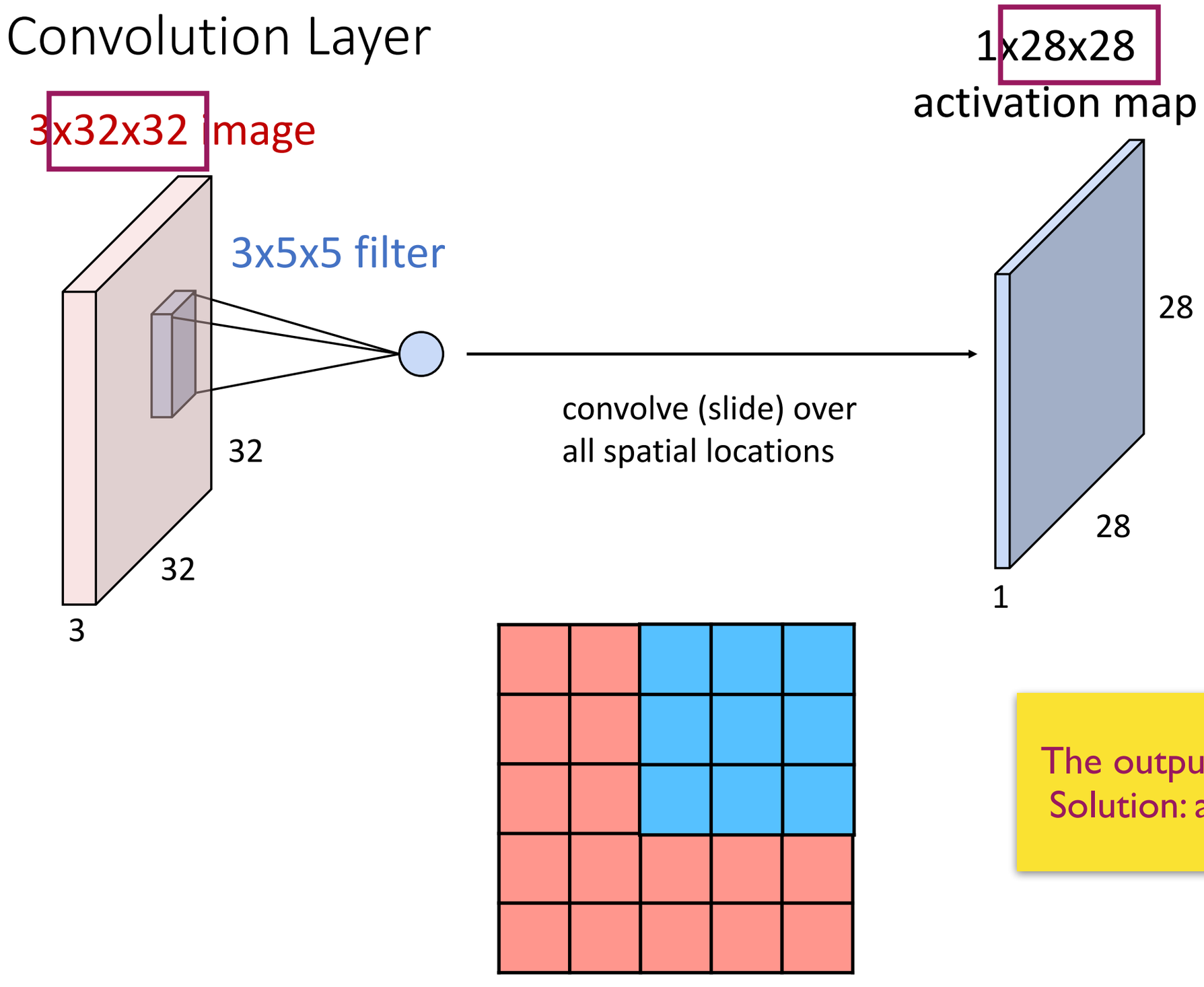
Padding


Convolution Layer
3x32x32 image
3x5x5 filter
32
32
3
convolve (slide) over
all spatial locations
1x28x28
activation map
28
28
1

Padding

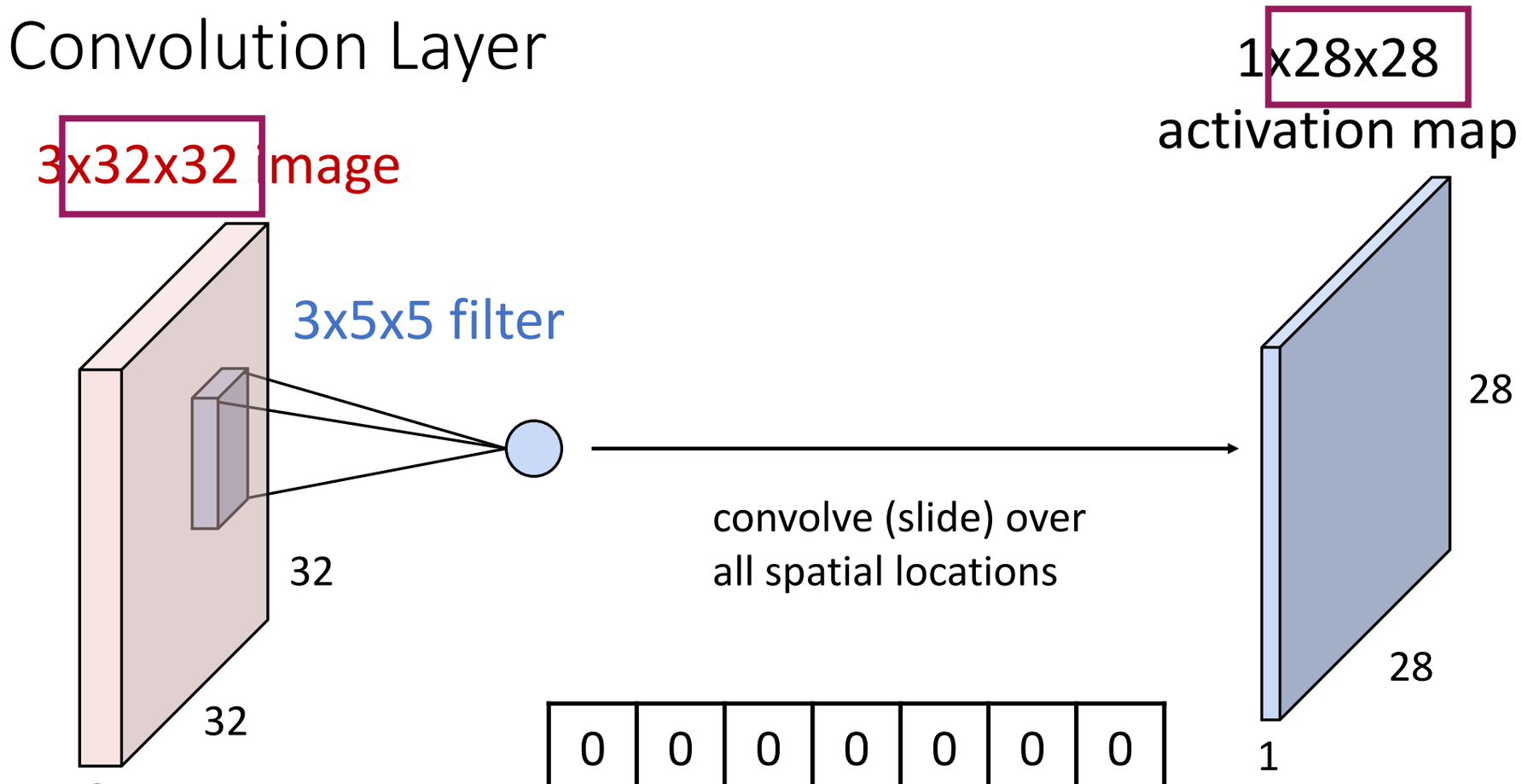

Convolution Layer
3x32x32 image
3x5x5 filter
32
32
3
convolve (slide) over
all spatial locations
1x28x28
activation map
28
28
1

Padding



The output size decreases!
Solution: adding on edges.

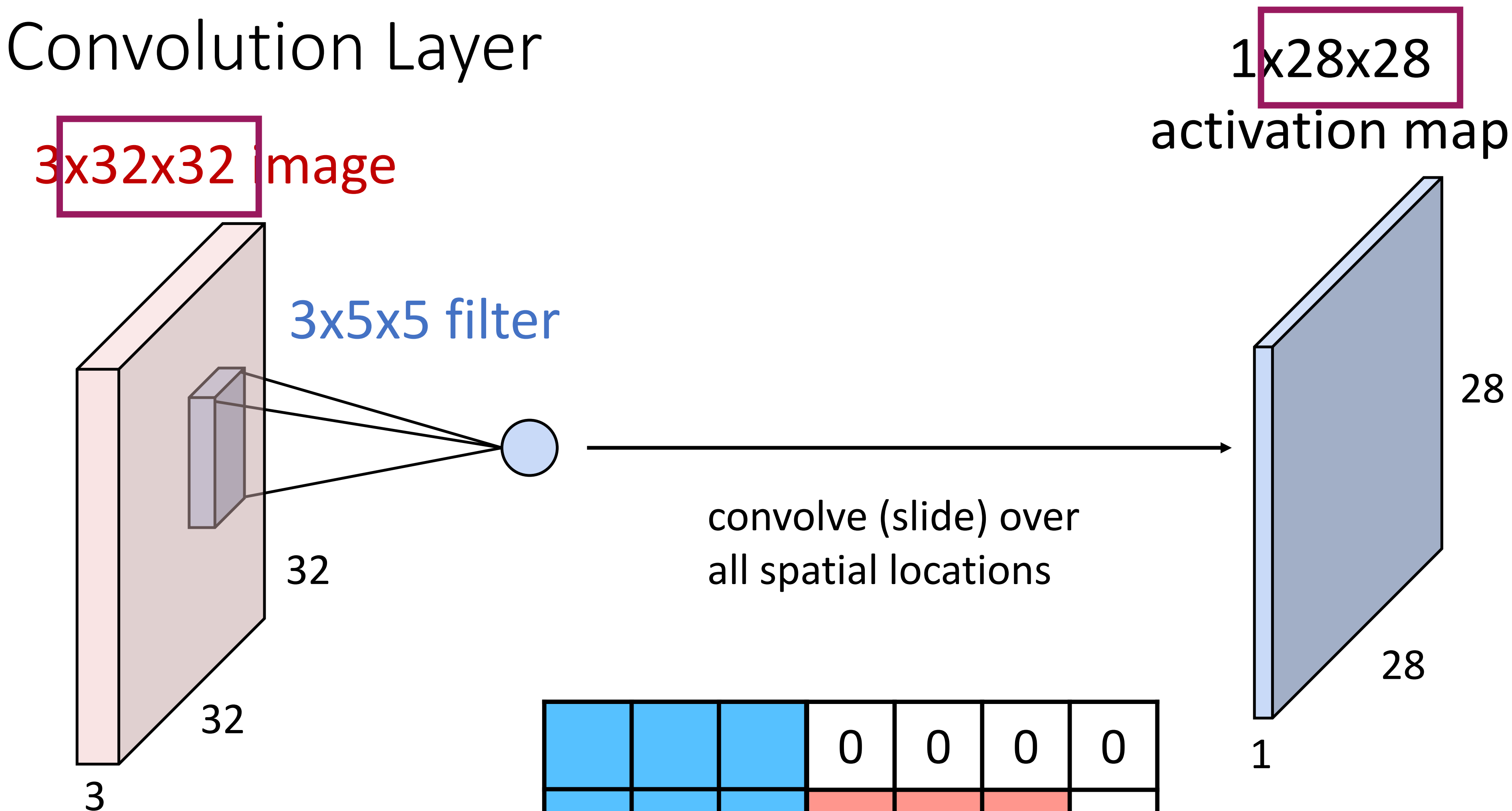
Padding

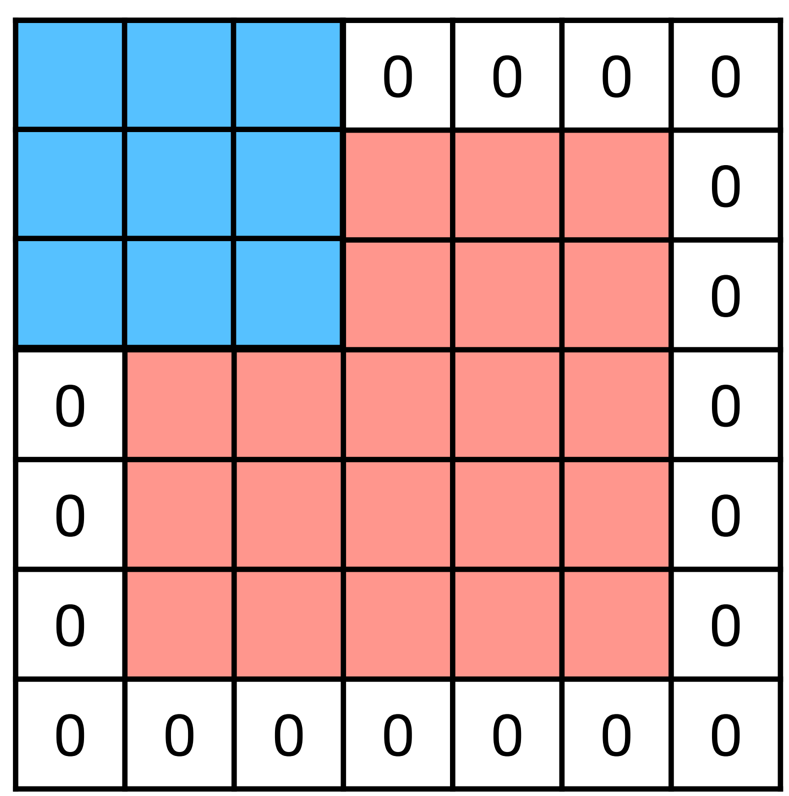


0	0	0	0	0	0	0
0						0
0						0
0						0
0						0
0						0
0	0	0	0	0	0	0

The output size decreases!
Solution: adding on edges.

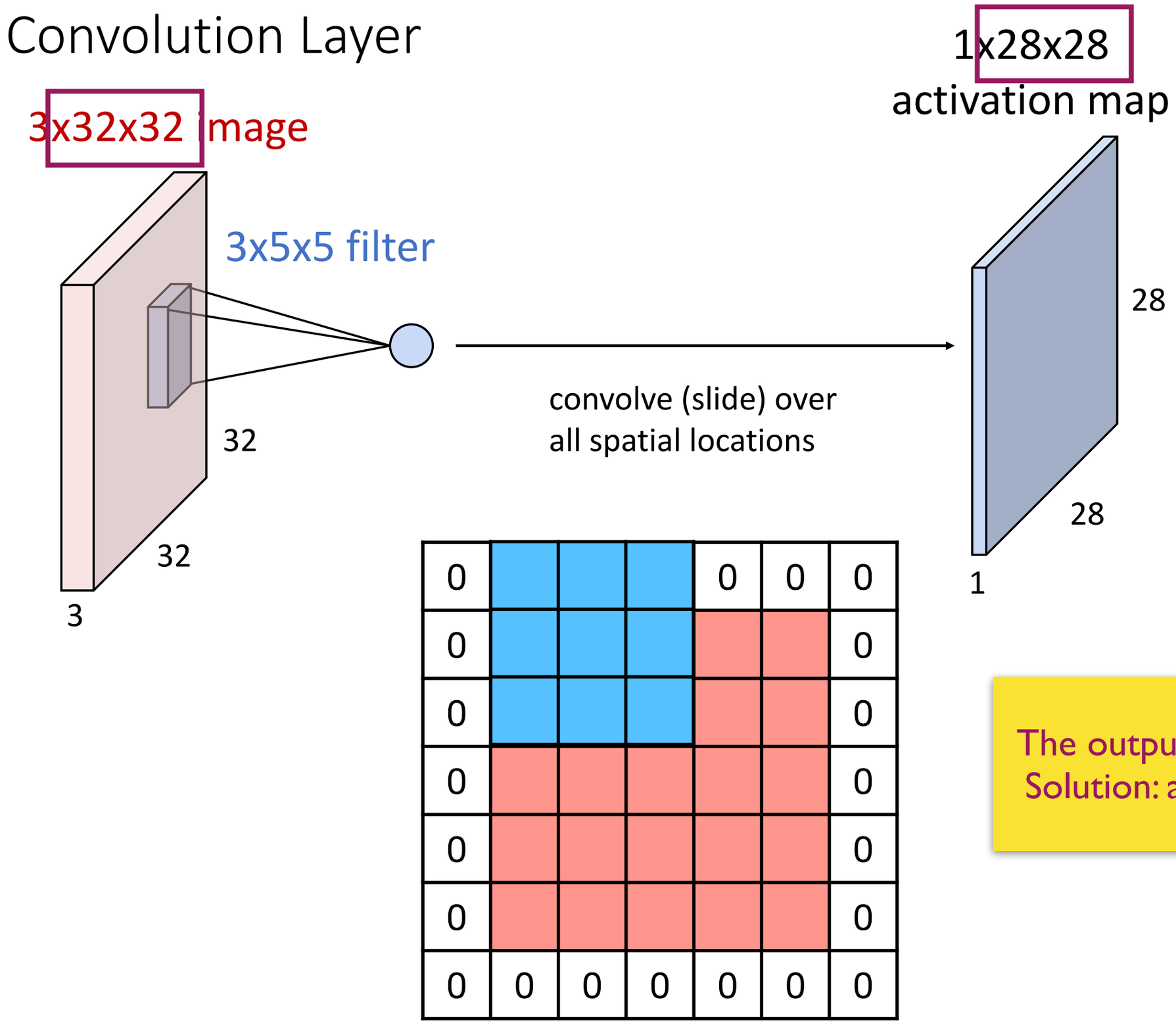
Padding





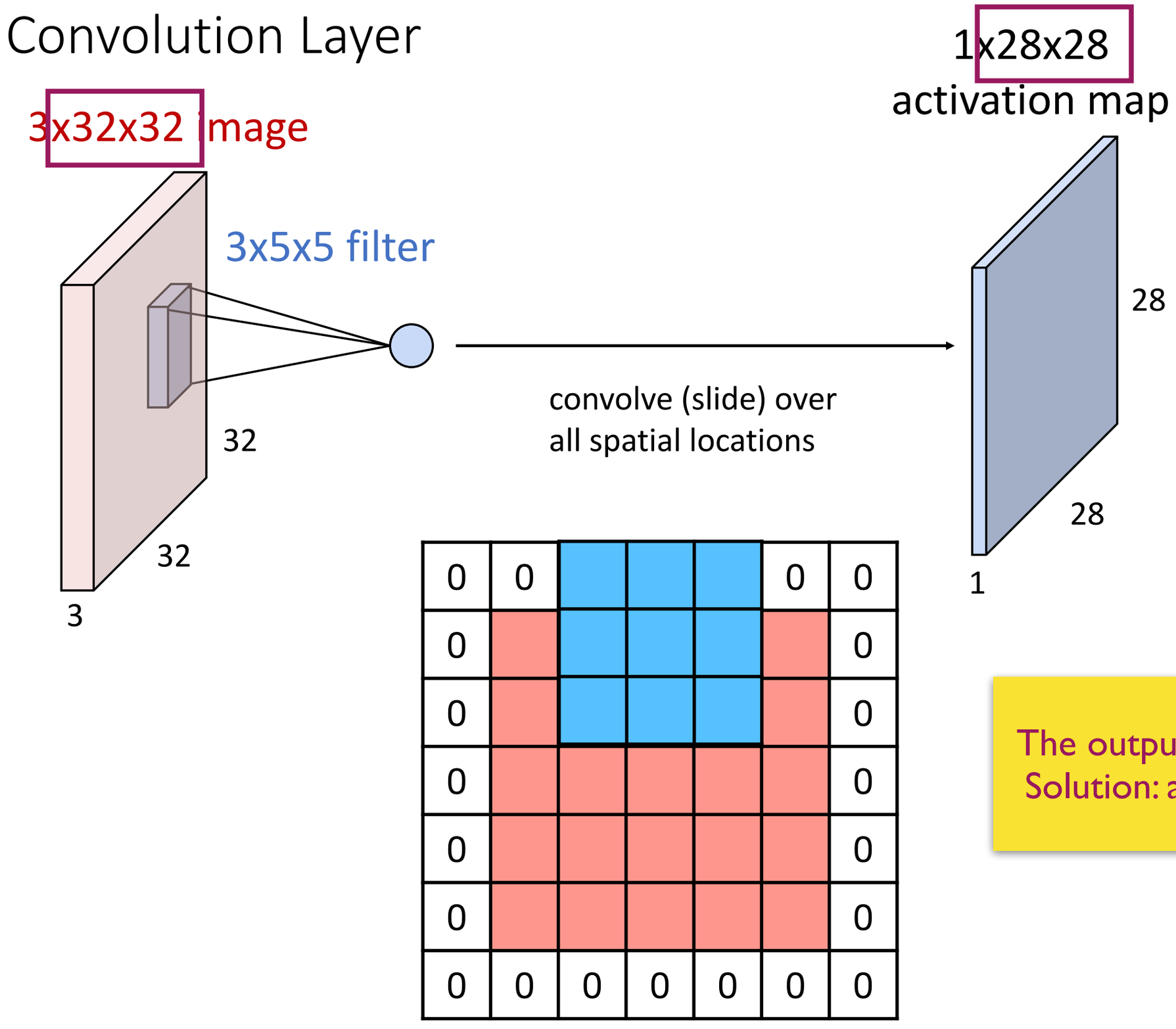
The output size decreases!
Solution: adding on edges.

Padding



The output size decreases!
Solution: adding on edges.

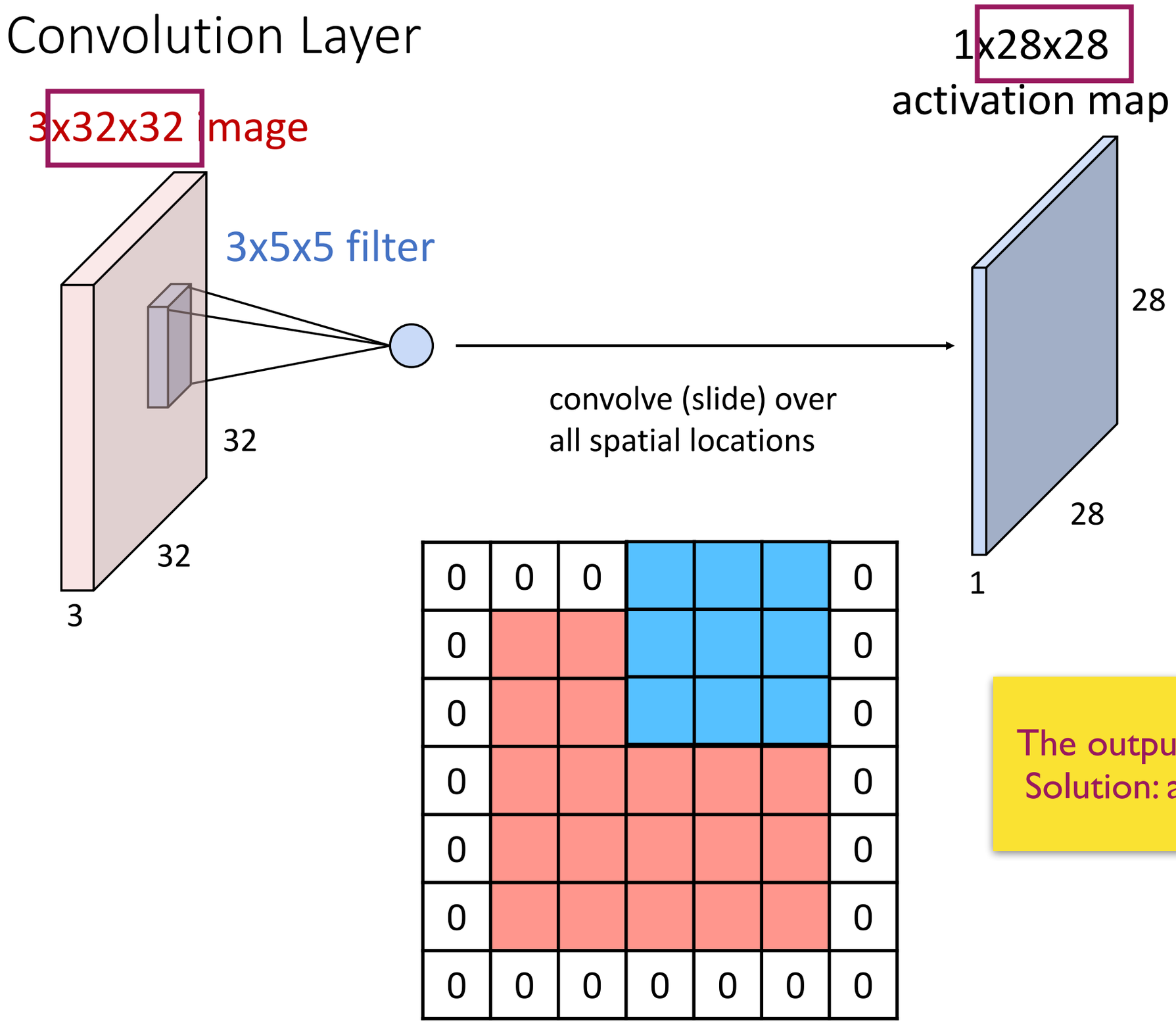
Padding



0	0				0	0
0						0
0						0
0						0
0						0
0						0
0	0	0	0	0	0	0

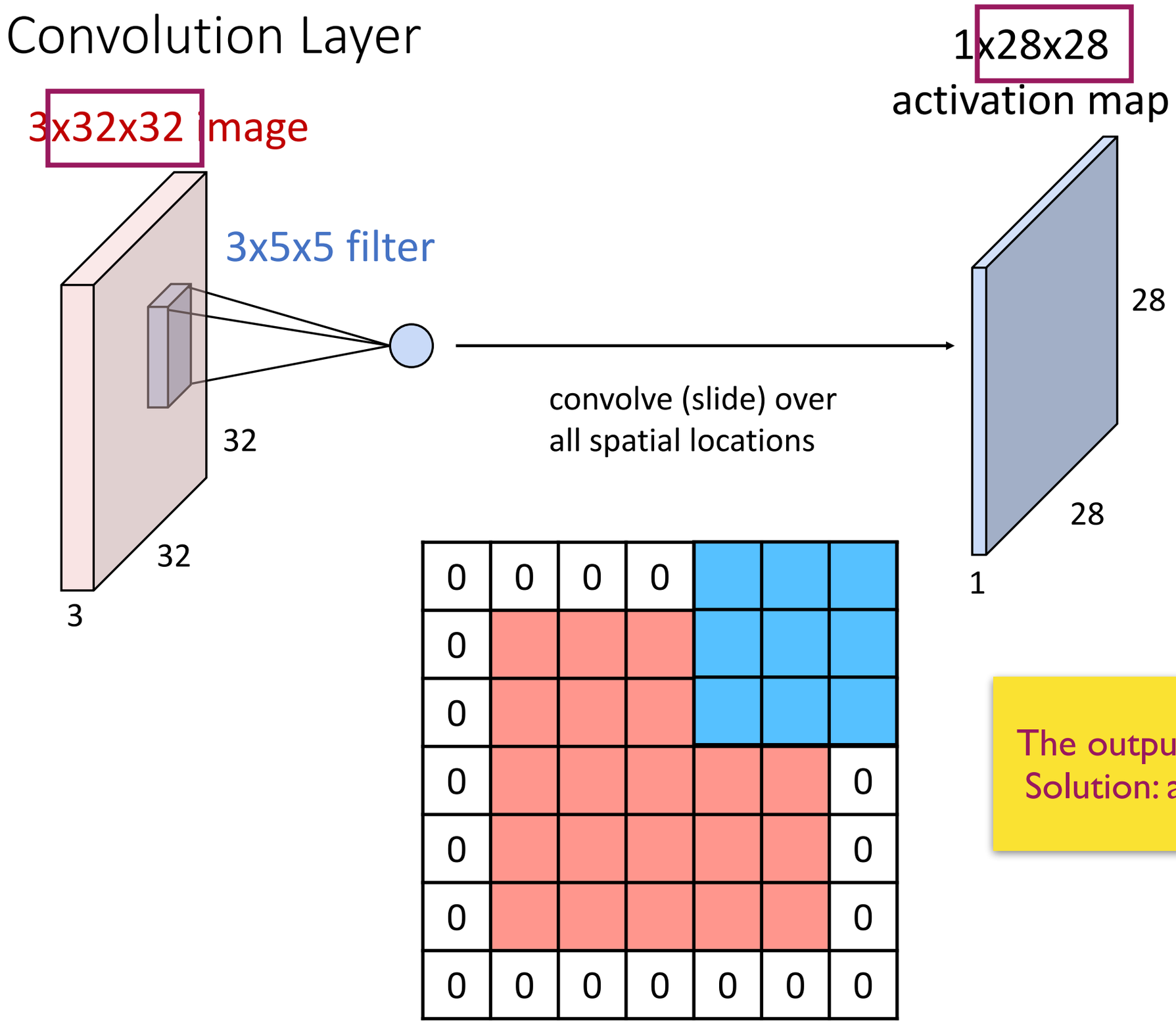
The output size decreases!
Solution: adding on edges.

Padding



The output size decreases!
Solution: adding on edges.

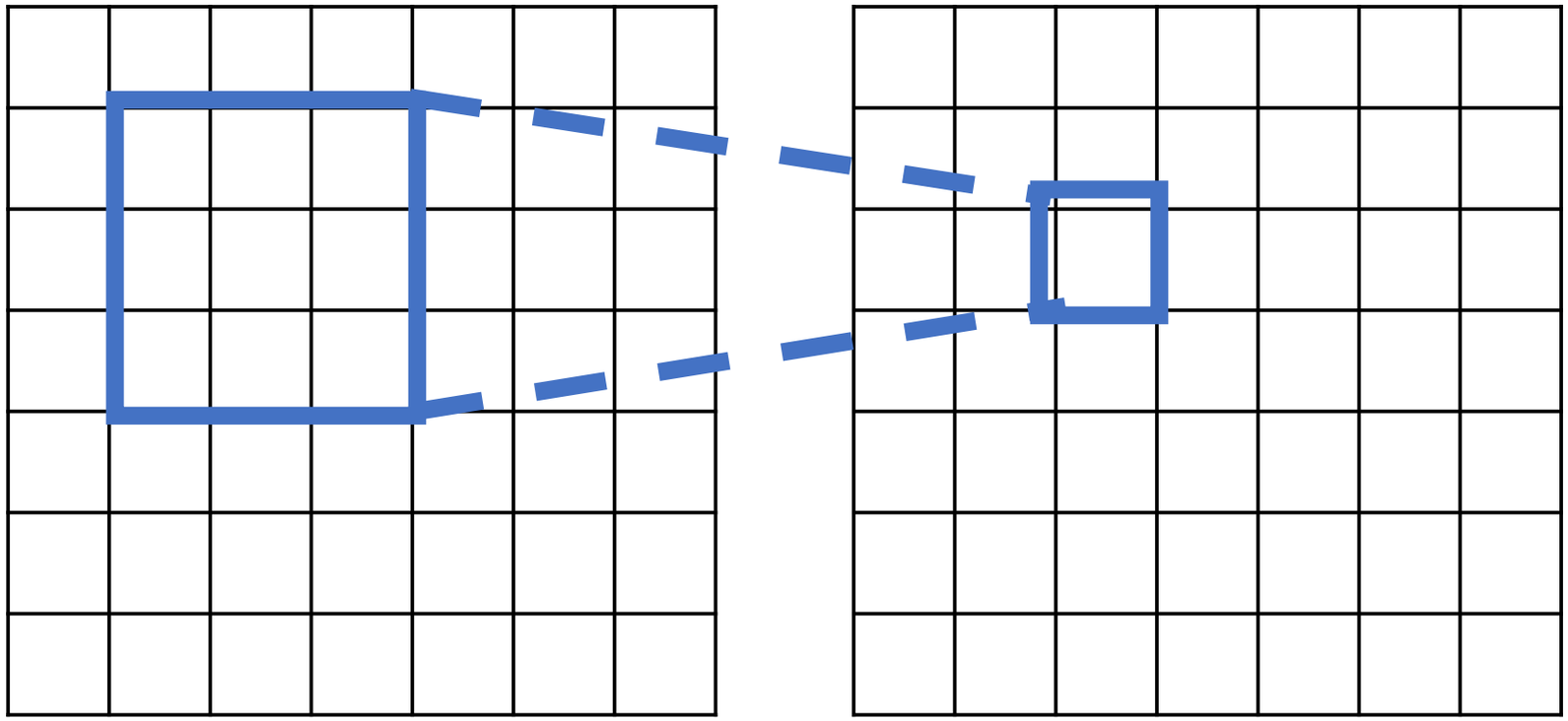
Padding



The output size decreases!
Solution: adding on edges.

Receptive Fields

For convolution with kernel size K, each element in the output depends on a K x K **receptive field** in the input

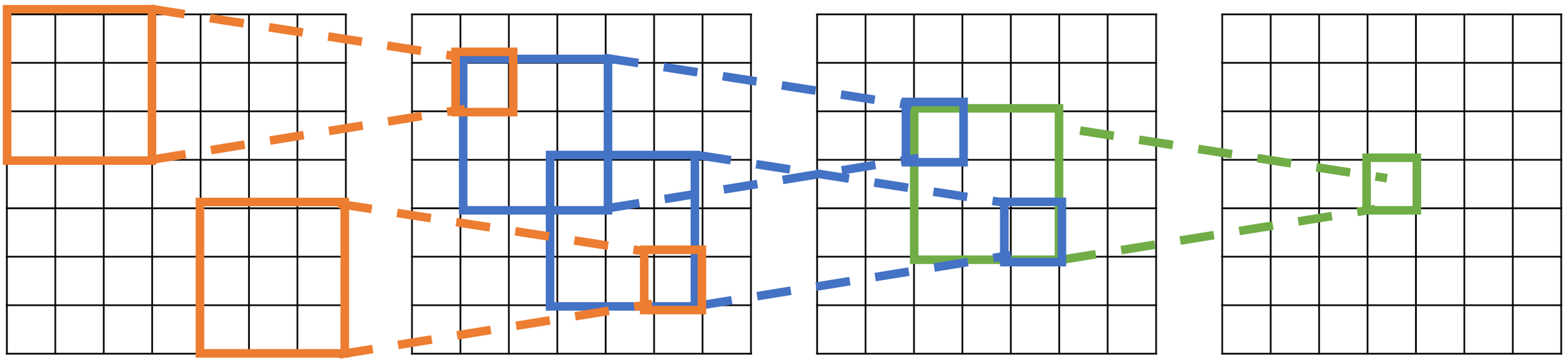


Input Output

Each unit in the output only corresponds a local area of the input.

Receptive Fields

Each successive convolution adds K – 1 to the receptive field size
With L layers the receptive field size is 1 + L * (K – 1)



Input

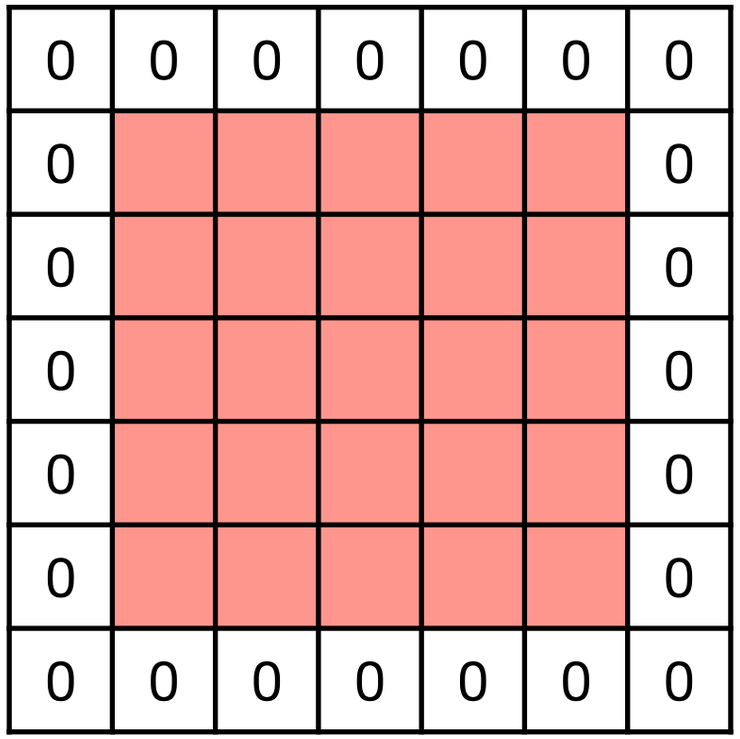
Output

Problem: For large images we need many layers for each output to “see” the whole image image

Want each output unit to combine the information of whole image.
Solution: downsample the input within the network.

Stride

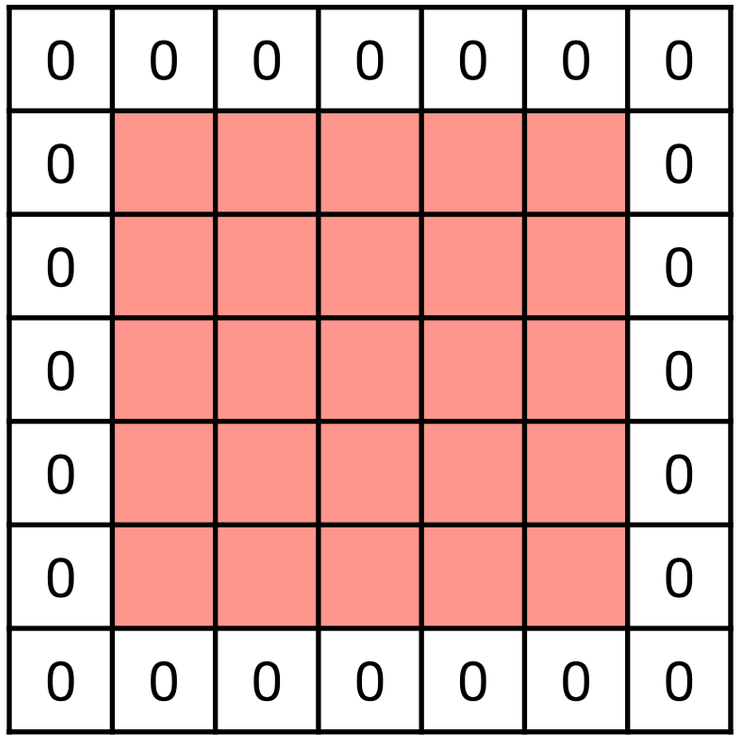
Stride: sliding the filter with a step size.
Large strides indicate sliding with large jumps.



Stride

Stride: sliding the filter with a step size.
Large strides indicate sliding with large jumps.

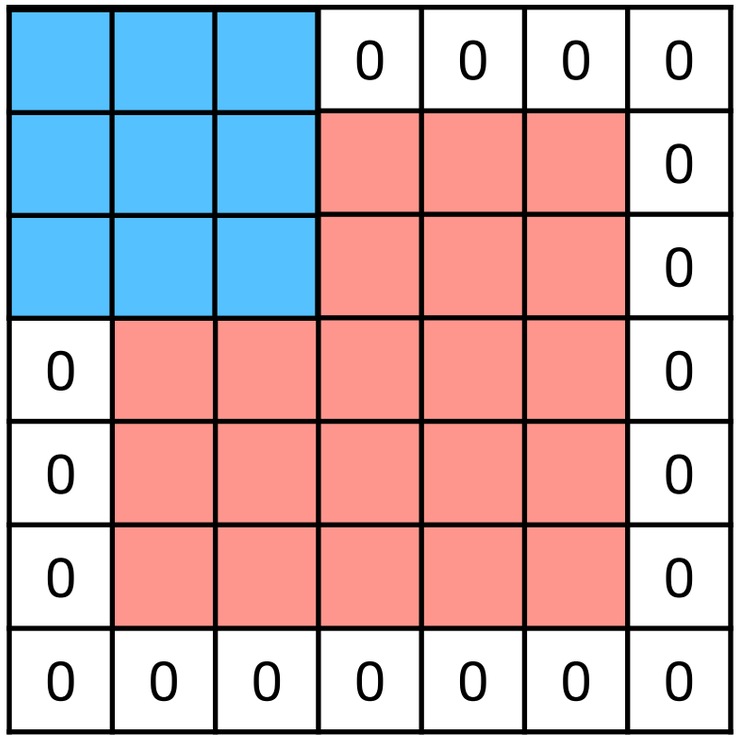
Stride = 1



Stride

Stride: sliding the filter with a step size.
Large strides indicate sliding with large jumps.

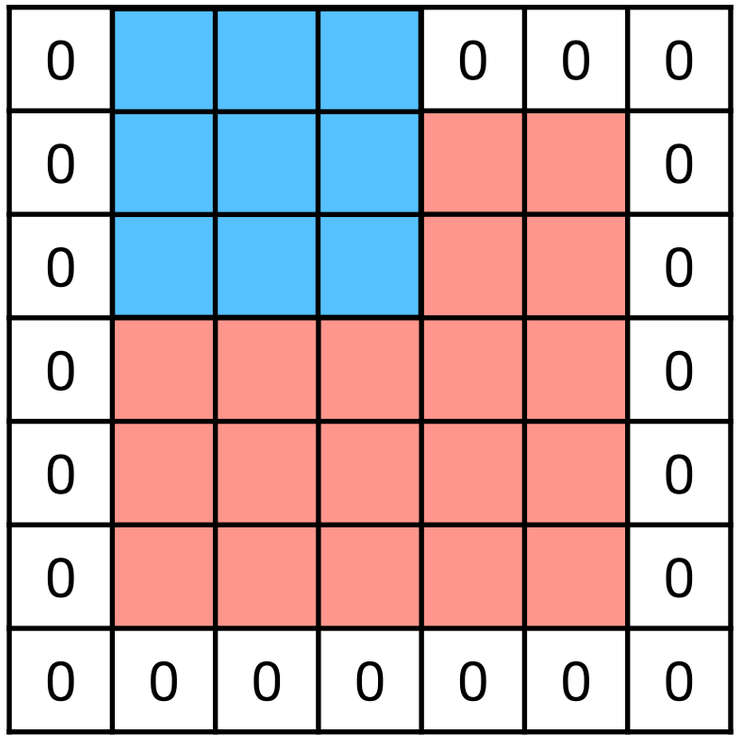
Stride = 1



Stride

Stride: sliding the filter with a step size.
Large strides indicate sliding with large jumps.

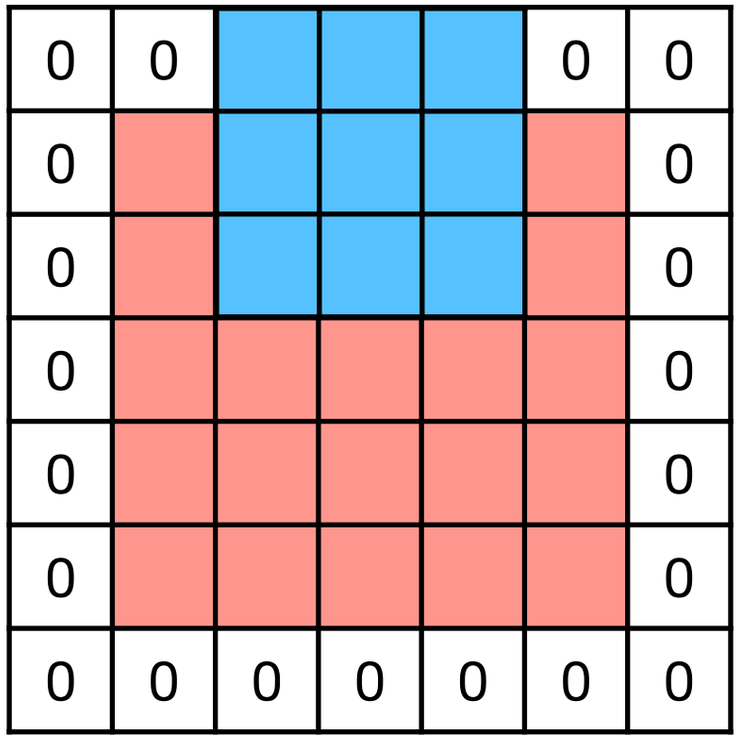
Stride = 1



Stride

Stride: sliding the filter with a step size.
Large strides indicate sliding with large jumps.

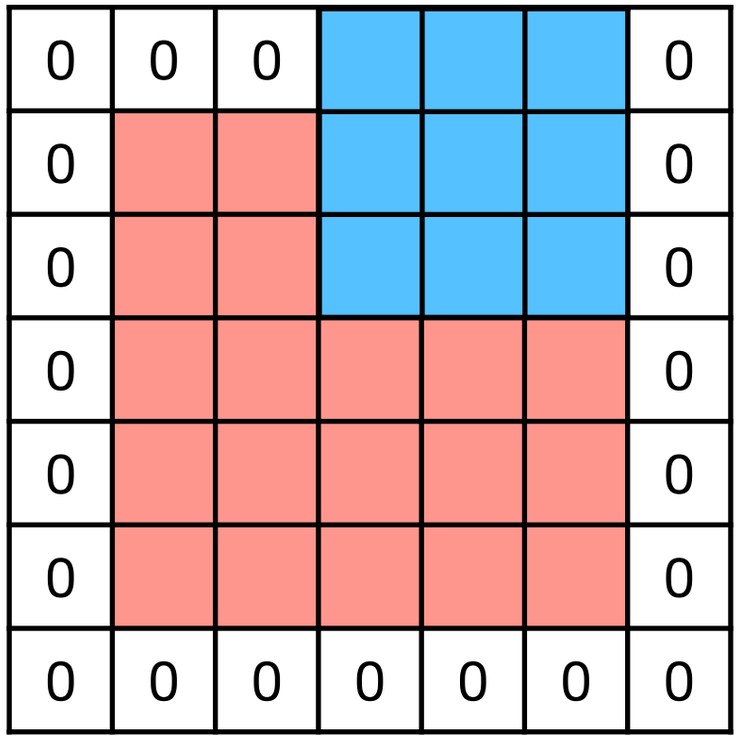
Stride = 1



Stride

Stride: sliding the filter with a step size.
Large strides indicate sliding with large jumps.

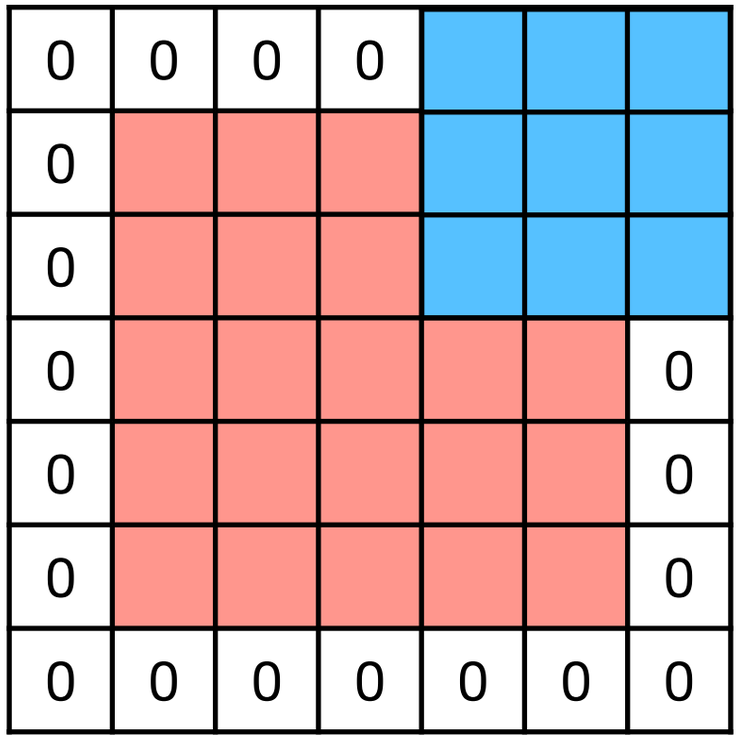
Stride = 1



Stride

Stride: sliding the filter with a step size.
Large strides indicate sliding with large jumps.

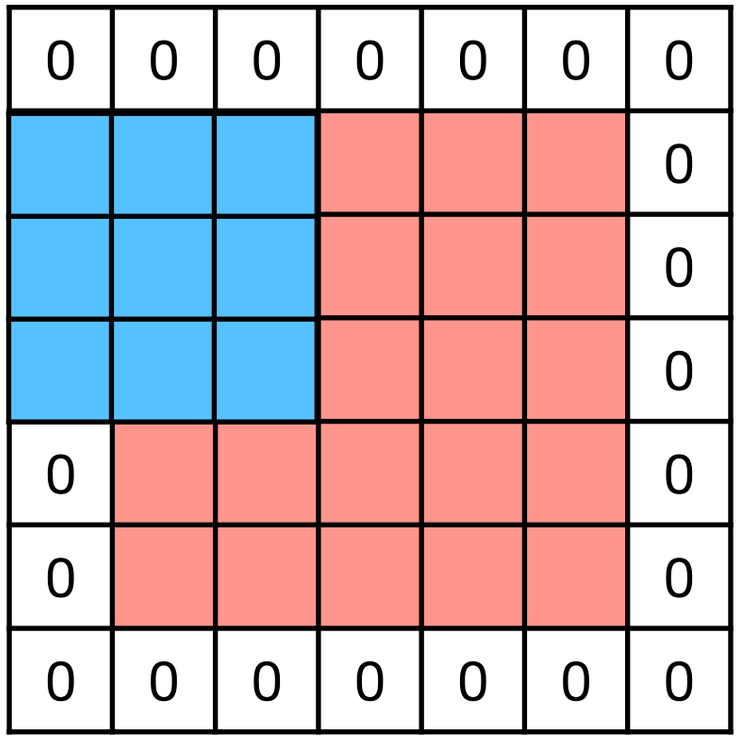
Stride = 1



Stride

Stride: sliding the filter with a step size.
Large strides indicate sliding with large jumps.

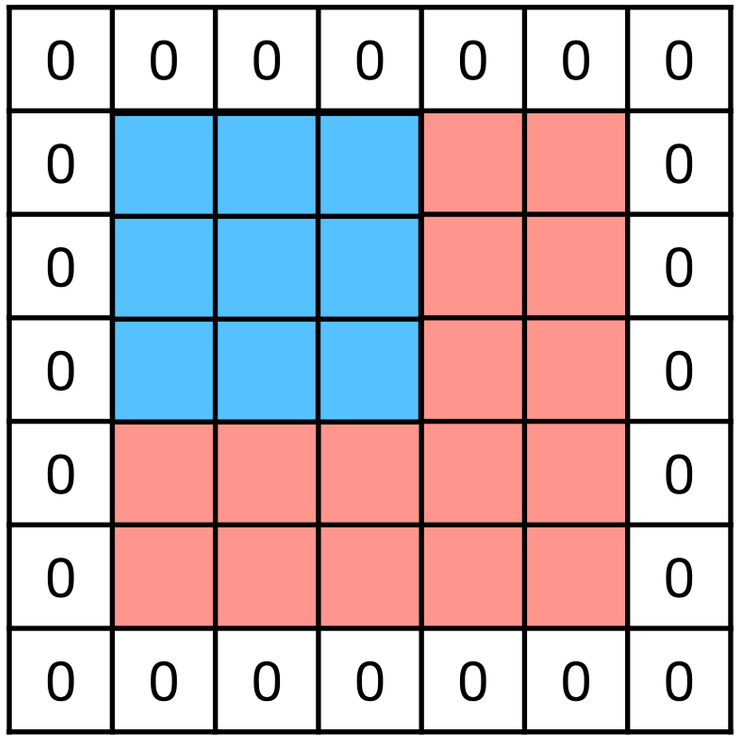
Stride = 1



Stride

Stride: sliding the filter with a step size.
Large strides indicate sliding with large jumps.

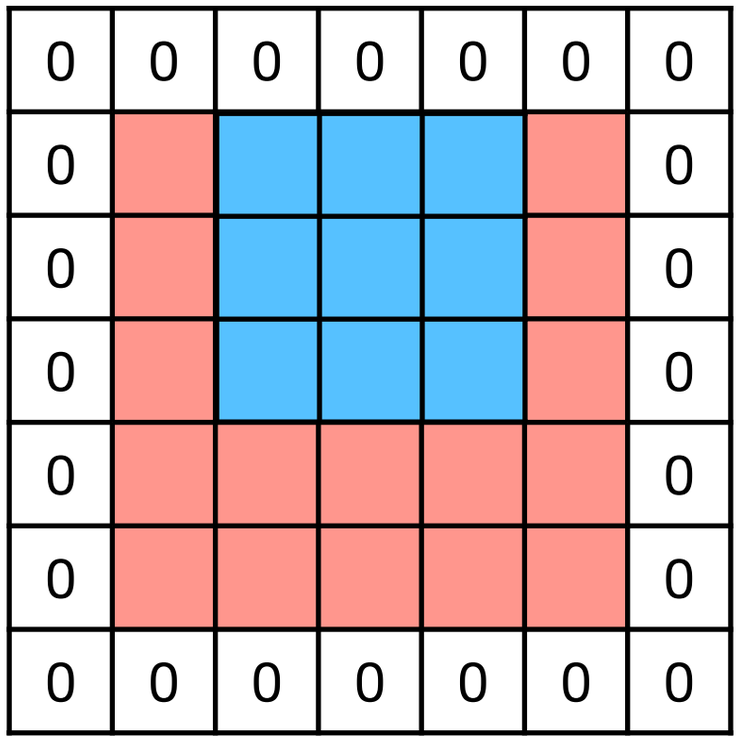
Stride = 1



Stride

Stride: sliding the filter with a step size.
Large strides indicate sliding with large jumps.

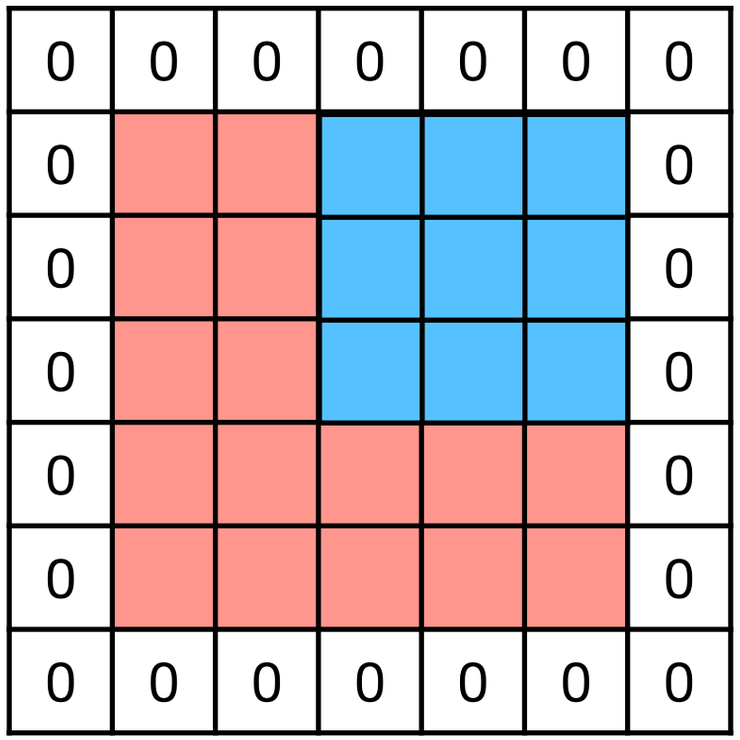
Stride = 1



Stride

Stride: sliding the filter with a step size.
Large strides indicate sliding with large jumps.

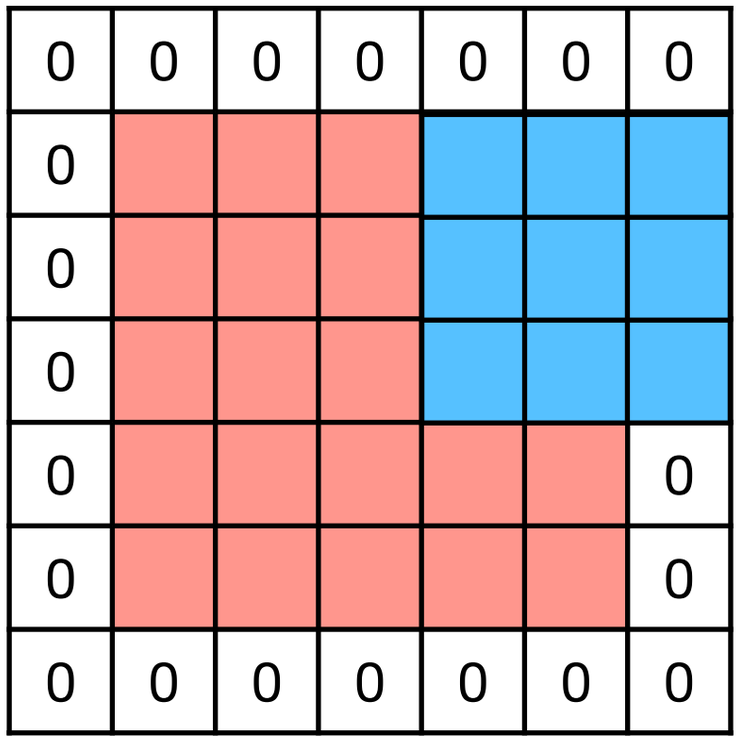
Stride = 1



Stride

Stride: sliding the filter with a step size.
Large strides indicate sliding with large jumps.

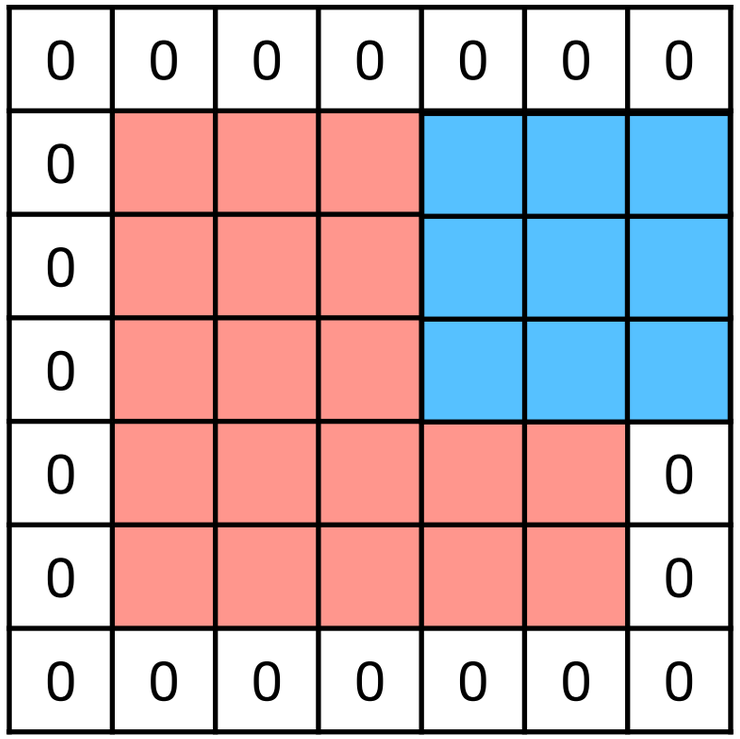
Stride = 1



Stride

Stride: sliding the filter with a step size.
Large strides indicate sliding with large jumps.

Stride = 1



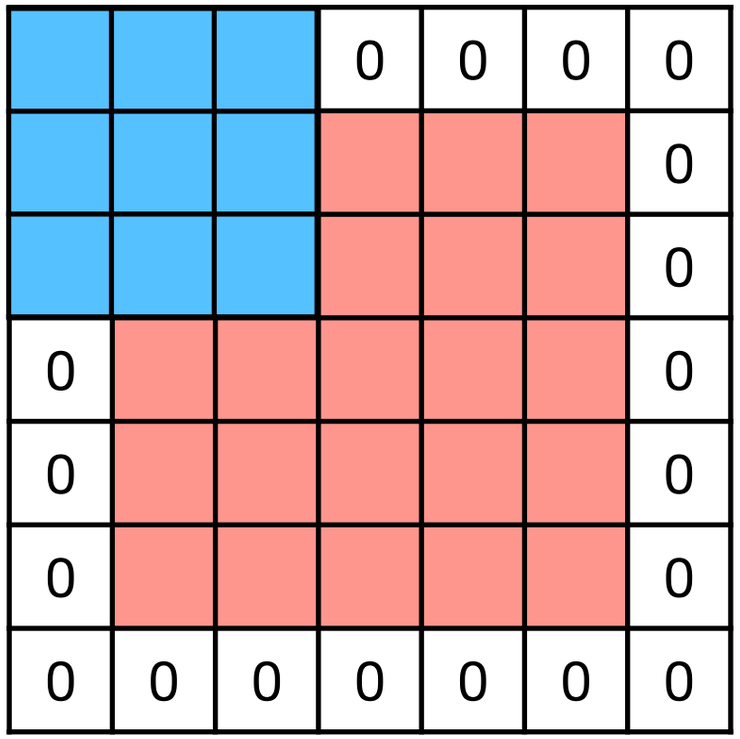
5*5 to 5*5

Stride

Stride: sliding the filter with a step size.
Large strides indicate sliding with large jumps.

Stride = 1

Stride = 2



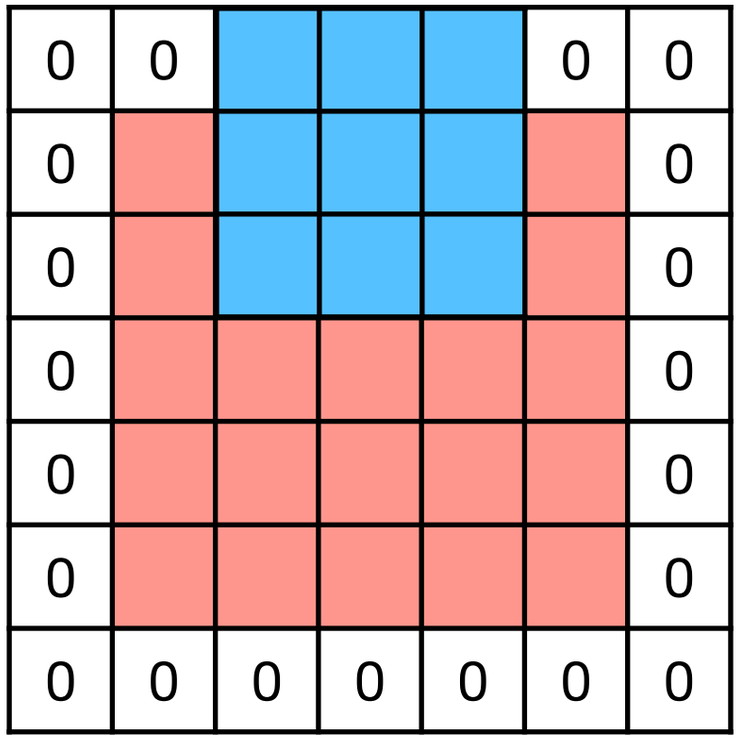
5*5 to 5*5

Stride

Stride: sliding the filter with a step size.
Large strides indicate sliding with large jumps.

Stride = 1

Stride = 2



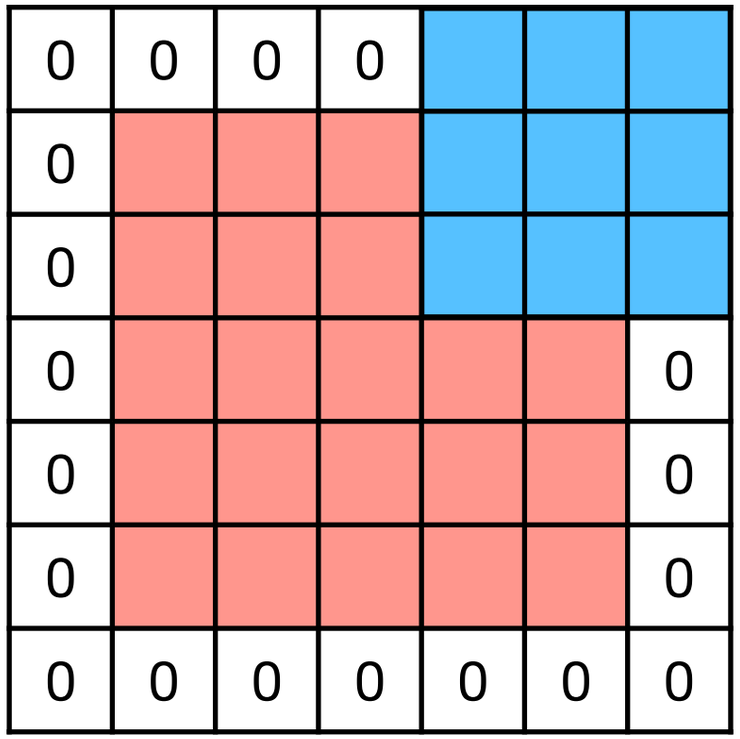
5*5 to 5*5

Stride

Stride: sliding the filter with a step size.
Large strides indicate sliding with large jumps.

Stride = 1

Stride = 2



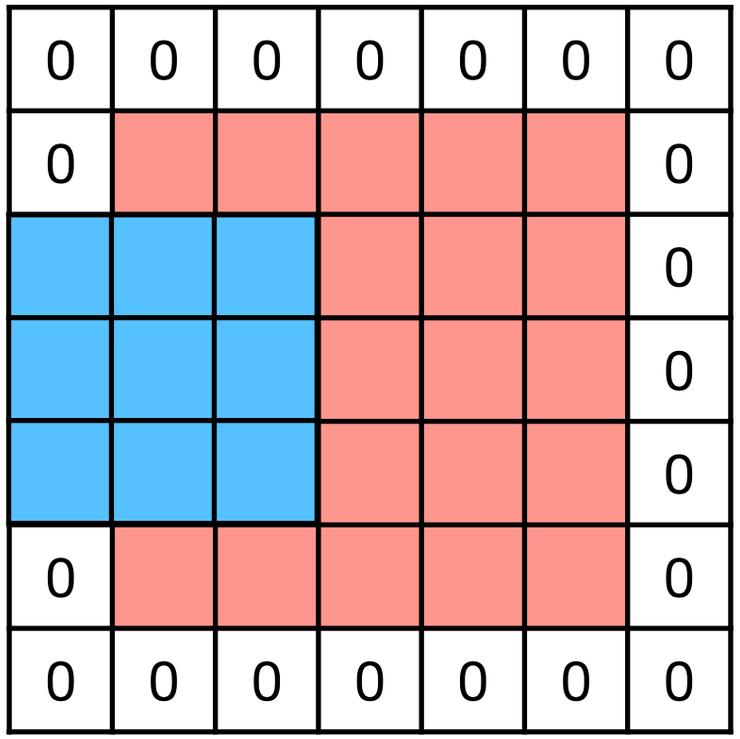
5*5 to 5*5

Stride

Stride: sliding the filter with a step size.
Large strides indicate sliding with large jumps.

Stride = 1

Stride = 2



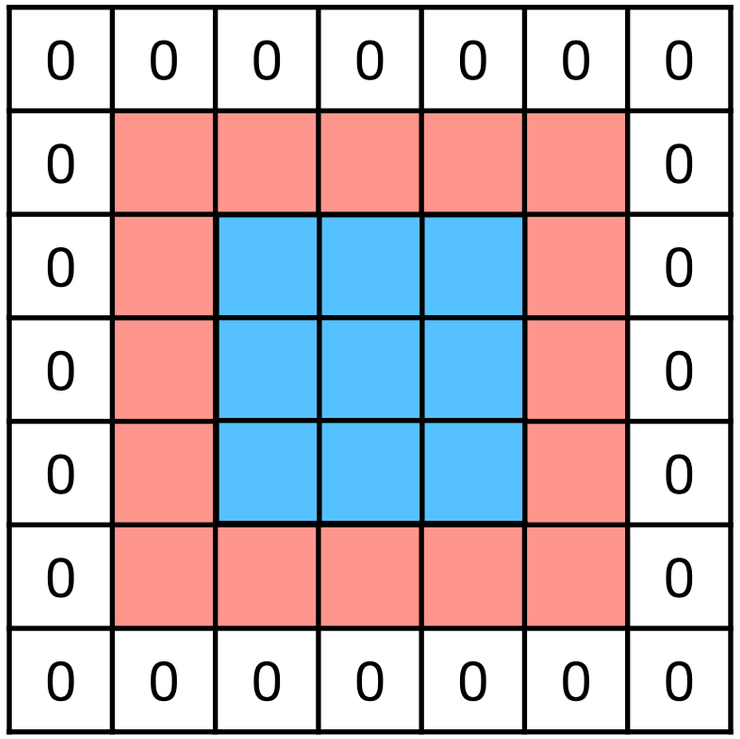
5*5 to 5*5

Stride

Stride: sliding the filter with a step size.
Large strides indicate sliding with large jumps.

Stride = 1

Stride = 2



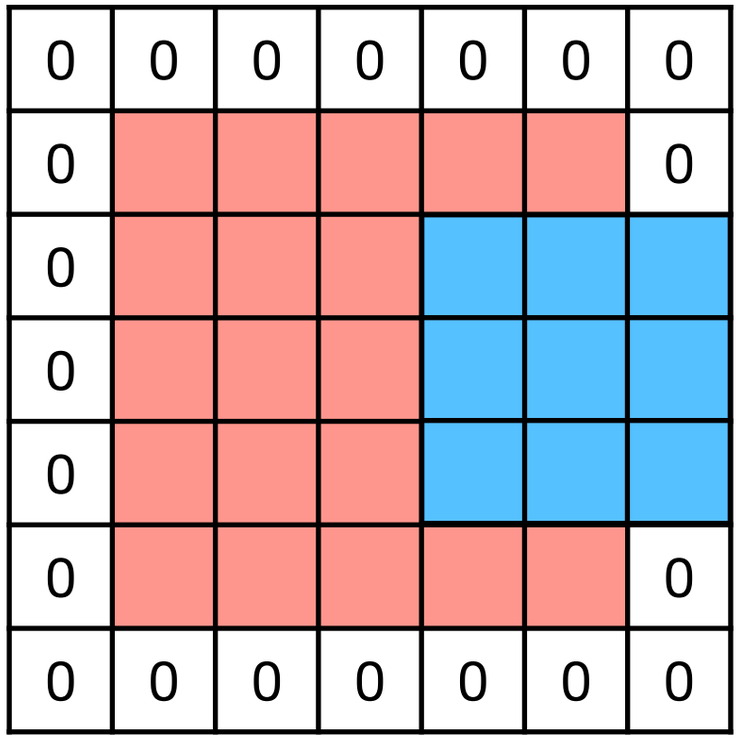
5*5 to 5*5

Stride

Stride: sliding the filter with a step size.
Large strides indicate sliding with large jumps.

Stride = 1

Stride = 2



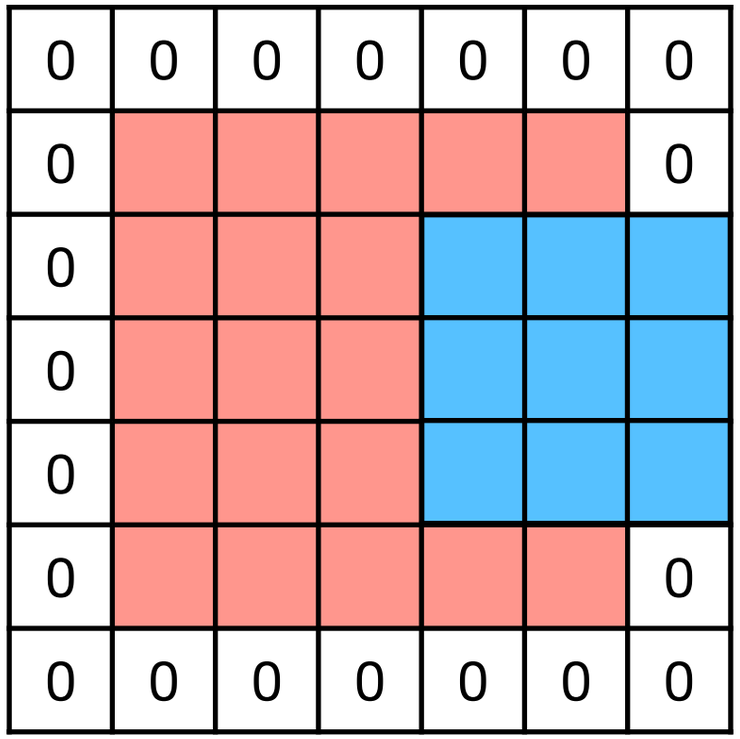
5*5 to 5*5

Stride

Stride: sliding the filter with a step size.
Large strides indicate sliding with large jumps.

Stride = 1

Stride = 2

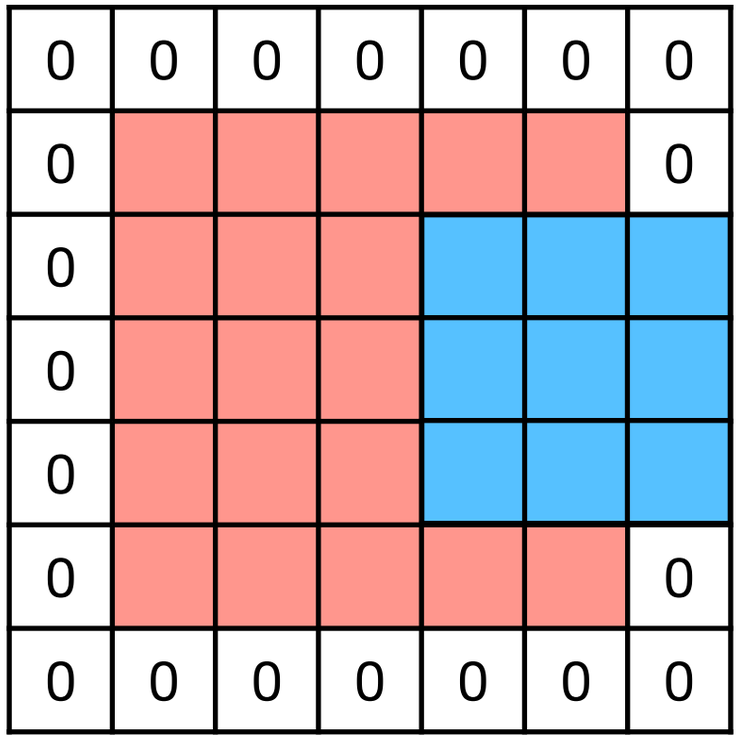


5*5 to 5*5

5*5 to 3*3

Stride

Stride: sliding the filter with a step size.
Large strides indicate sliding with large jumps.



Stride = 1 **5*5 to 5*5**

Stride = 2 **5*5 to 3*3**

Large stride downsamples the input.
Any other way to downsample?

Pooling Layer

Max Pooling

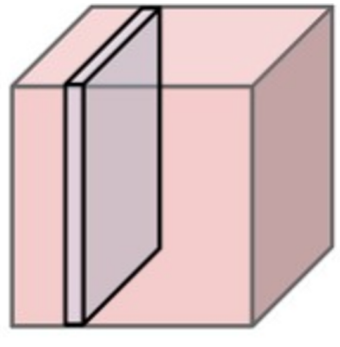
Single depth slice

x

y

1	1	2	4
5	**6**	7	**8**
3	2	1	0
1	2	3	**4**

64 x 224 x 224



Max pooling with 2x2 kernel size and stride 2

6	8
3	4

Introduces **invariance** to small spatial shifts

No learnable parameters!

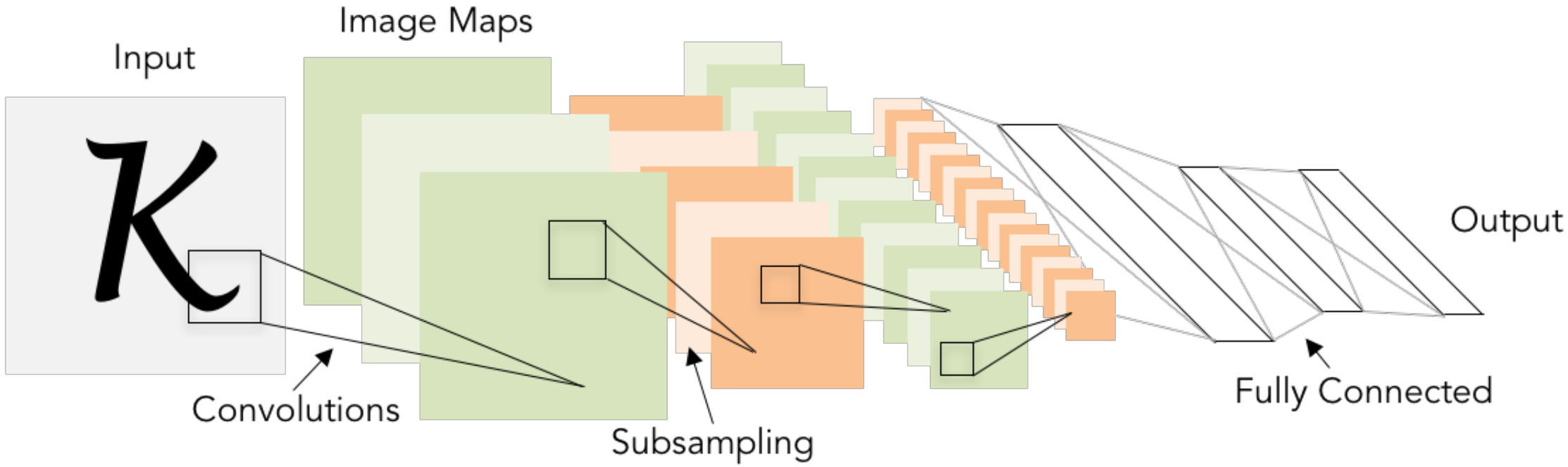
Convolutional Networks

PROC. OF THE IEEE, NOVEMBER 1998

Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

Example: LeNet-5



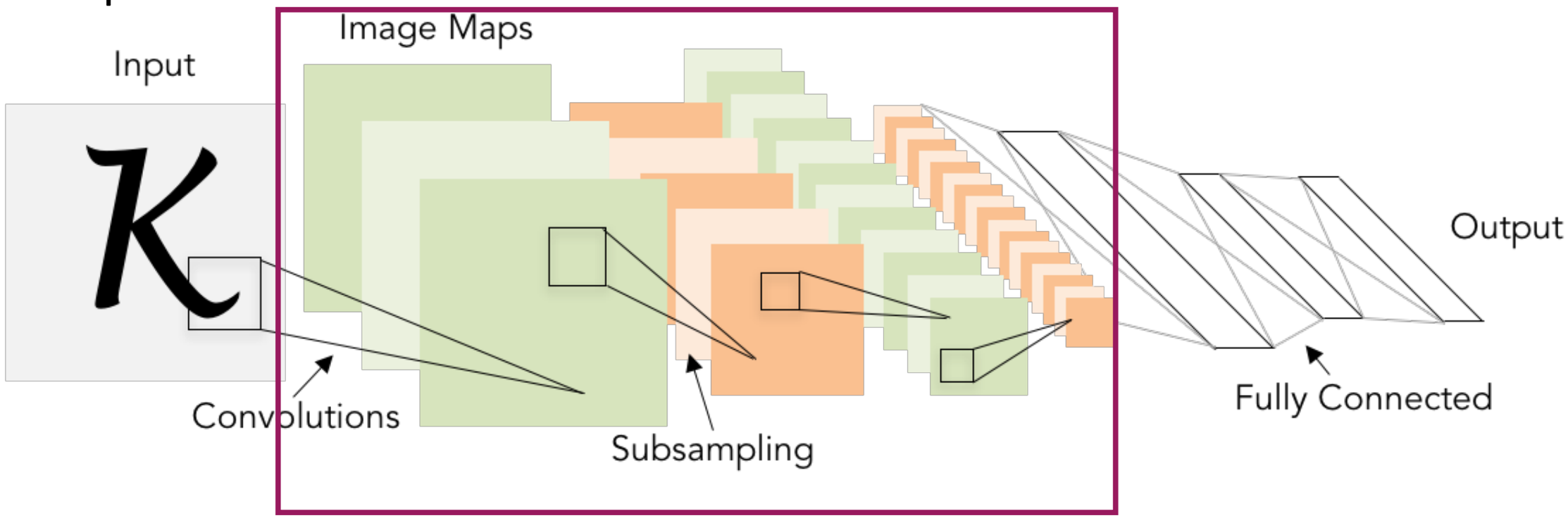
Convolutional Networks

PROC. OF THE IEEE, NOVEMBER 1998

Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

Example: LeNet-5



convolutional layers

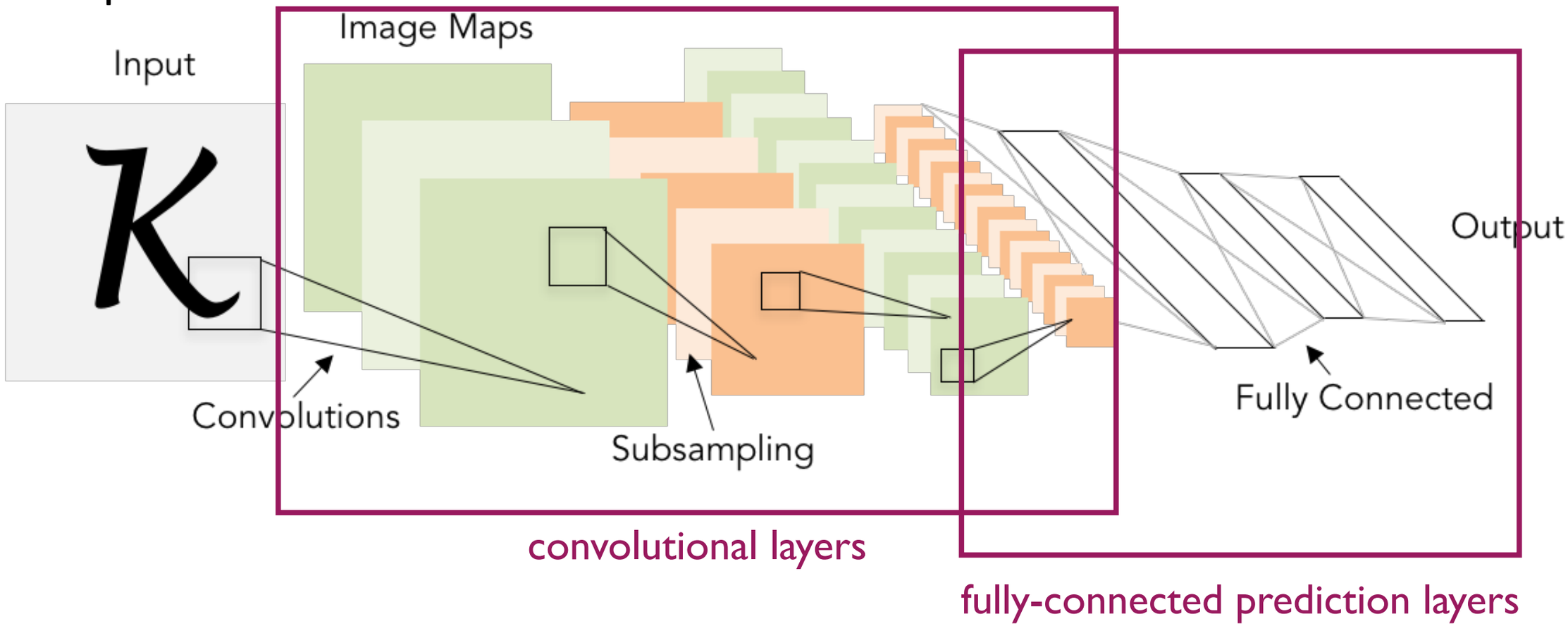
Convolutional Networks

PROC. OF THE IEEE, NOVEMBER 1998

Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

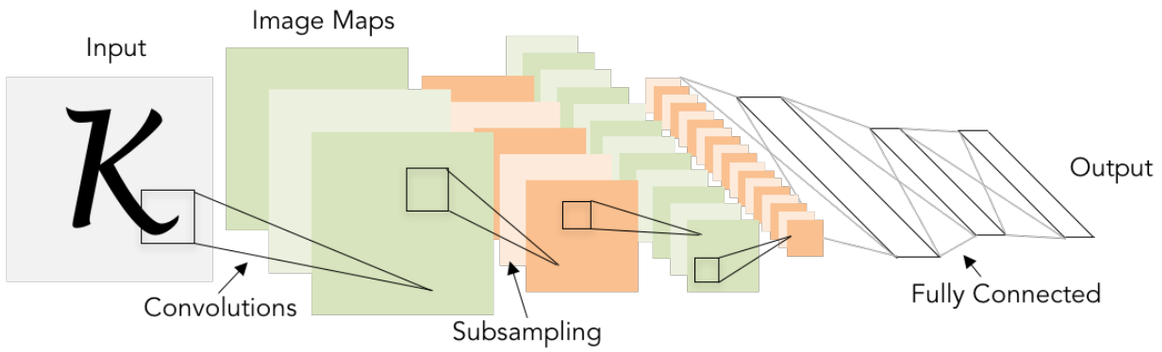
Example: LeNet-5



LeNet-5

Example: LeNet-5

Layer	Output Size	Weight Size
Input	1 x 28 x 28	
Conv (C_{out}=20, K=5, P=2, S=1)	20 x 28 x 28	20 x 1 x 5 x 5
ReLU	20 x 28 x 28	
MaxPool(K=2, S=2)	20 x 14 x 14	
Conv (C_{out}=50, K=5, P=2, S=1)	50 x 14 x 14	50 x 20 x 5 x 5
ReLU	50 x 14 x 14	
MaxPool(K=2, S=2)	50 x 7 x 7	
Flatten	2450	
Linear (2450 -> 500)	500	2450 x 500
ReLU	500	
Linear (500 -> 10)	10	500 x 10



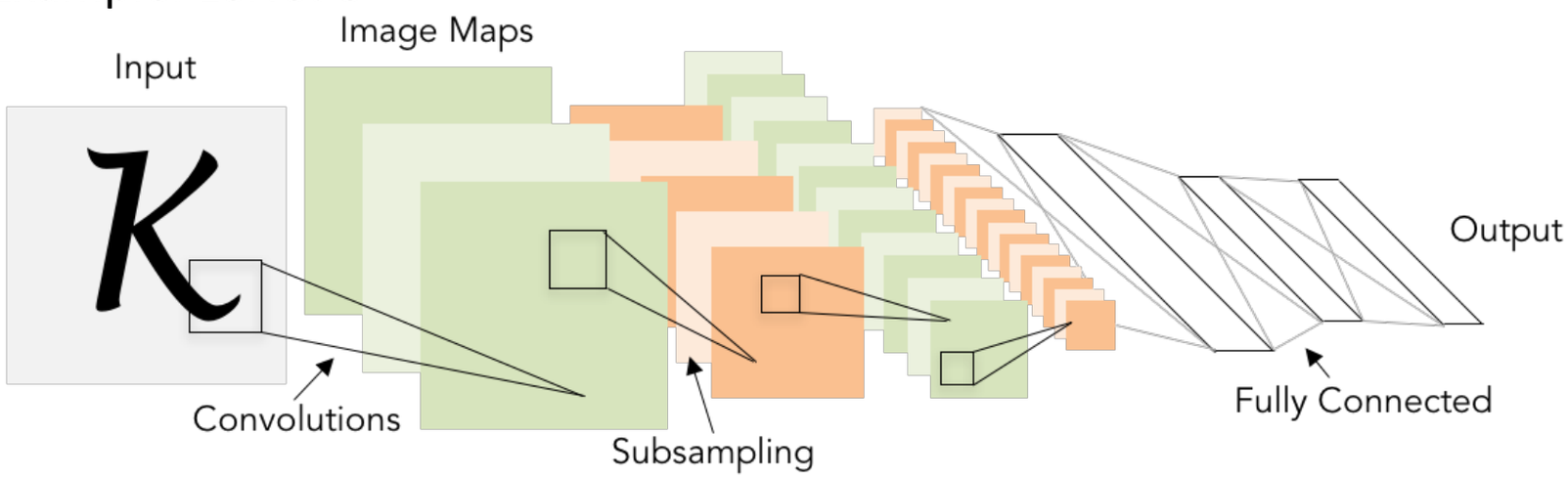
As we go through the network:

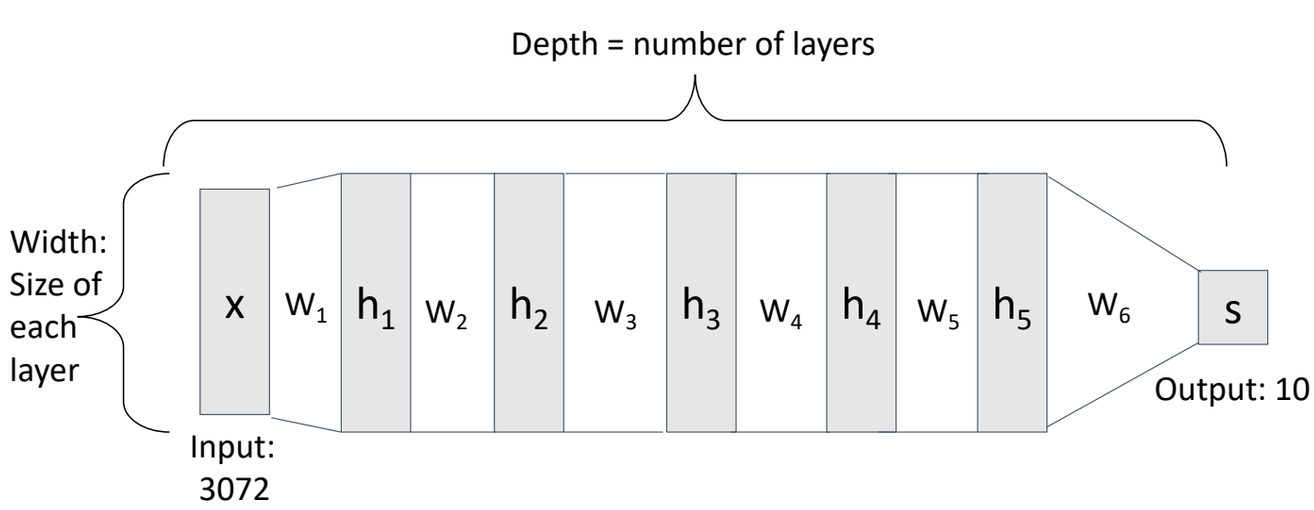
Spatial size **decreases**
(using pooling or strided conv)

Number of channels **increases**
(total “volume” is preserved!)

How Can We Go Deep?

Example: LeNet-5



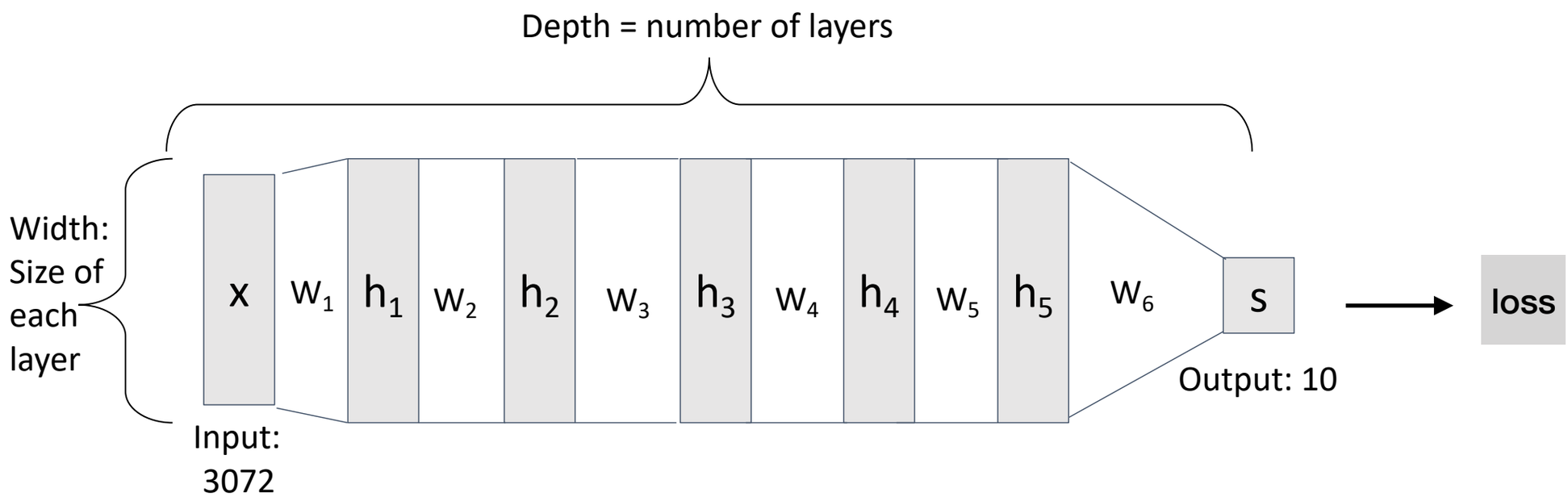


$$s = W_6 \max(0, W_5 \max(0, W_4 \max(0, W_3 \max(0, W_2 \max(0, W_1 x)))))$$

Why not just add more hidden layers?

Back Propagation & Gradient Vanishing

forward pass



$$s = W_6 \max(0, W_5 \max(0, W_4 \max(0, W_3 \max(0, W_2 \max(0, W_1 x)))))$$

backward pass

SGD & BP lies at the heart of NN learning.
In general, for backward pass, the gradient of bottom layers is much smaller than the gradient of top layers.
For deep NN, the gradient is difficult to be passed to the bottom layers.

Deep Learning

A fast learning algorithm for deep belief nets *

Geoffrey E. Hinton and **Simon Osindero**
Department of Computer Science University of Toronto
10 Kings College Road
Toronto, Canada M5S 3G4
{hinton, osindero}@cs.toronto.edu

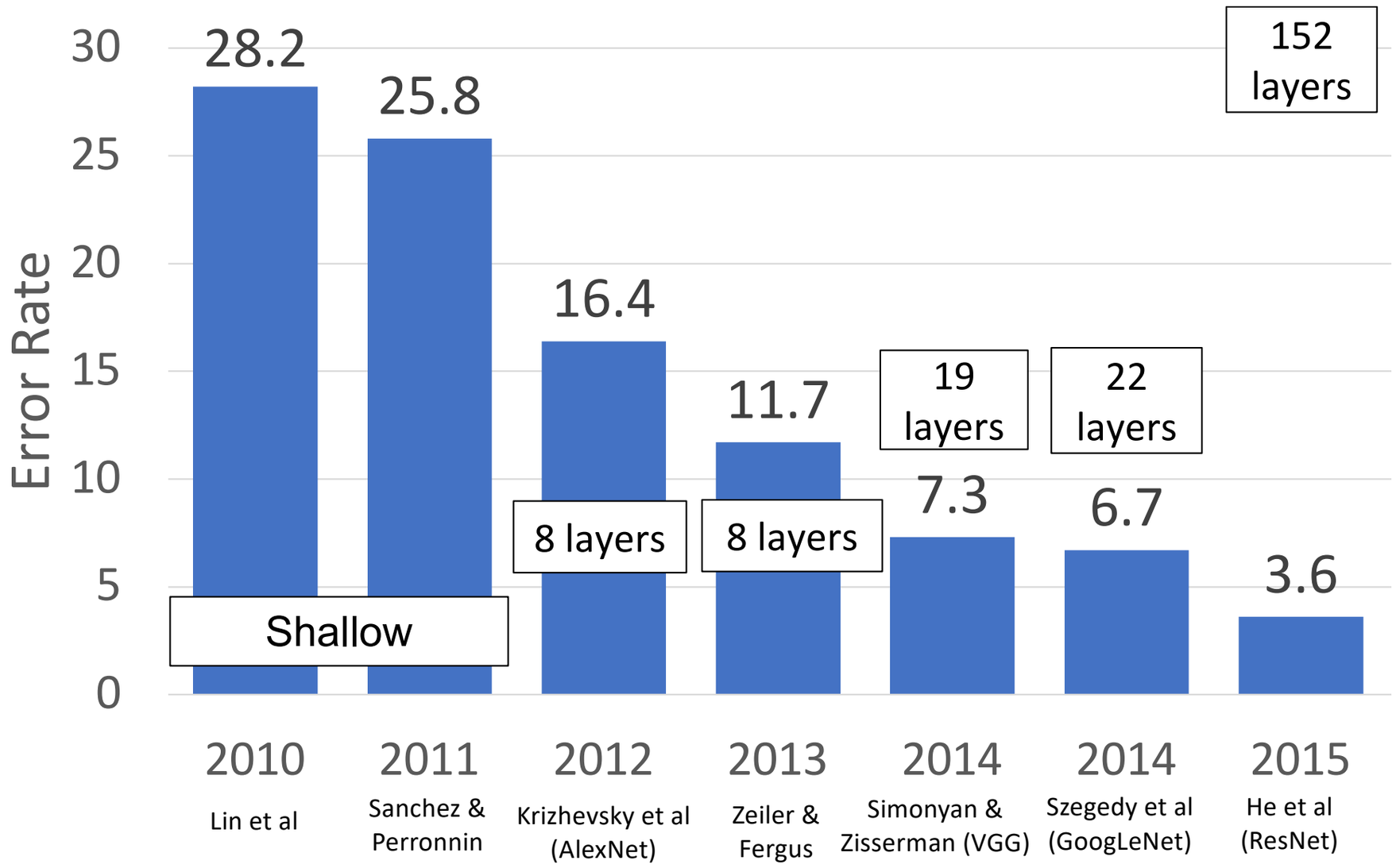
Yee-Whye Teh
Department of Computer Science
National University of Singapore
3 Science Drive 3, Singapore, 117543
tehyw@comp.nus.edu.sg

- In 2006, Hinton proposed a pre-training then BP method to train 4-layer NNs.

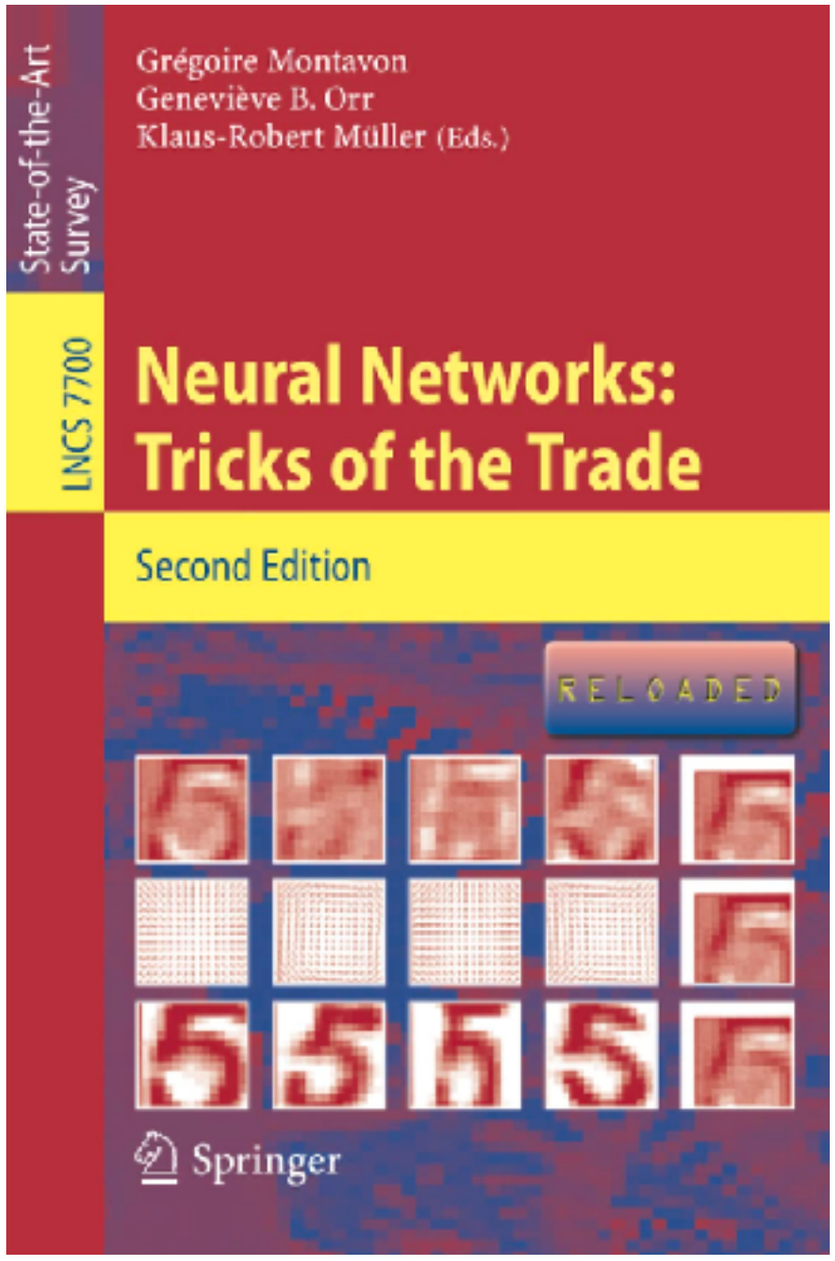
- First application in speech recognition.

- Explosion from 2012: ImageNet challenge for image classification.

The ImageNet Challenge



More Tricks to Work Better



Machine Learning: V

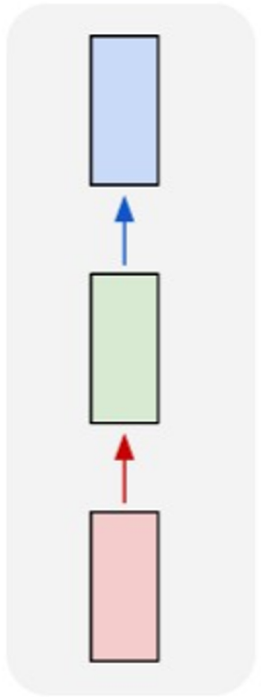
- NN to process sequential data
  - Recurrent NN

- Generative NN models
  - Autoencoders
  - GANs

- Take-home messages

Sequential Learning Problems

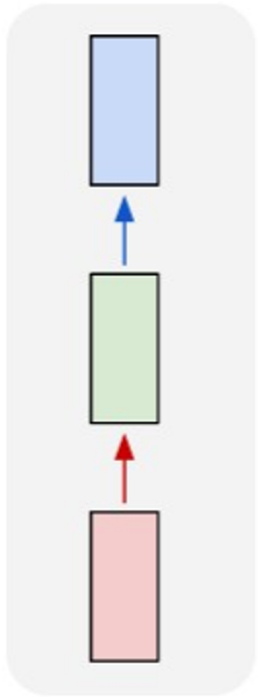
one-to-one



e.g.
image
classification

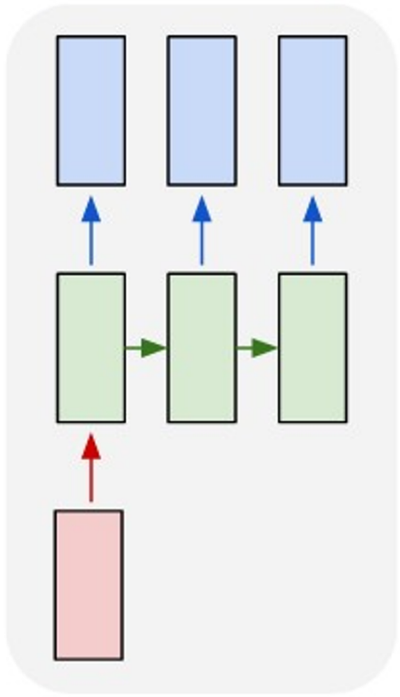
Sequential Learning Problems

one-to-one



e.g.
image
classification

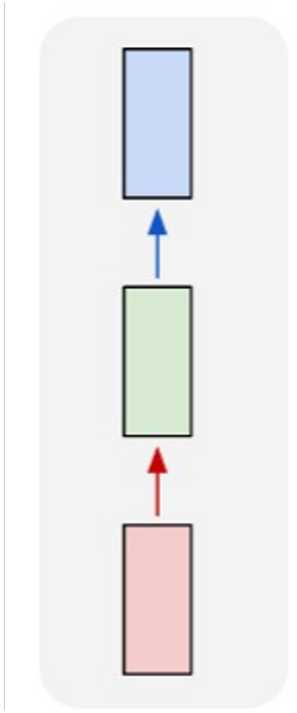
one-to-many



e.g.
image captioning:
image ->sequence of words

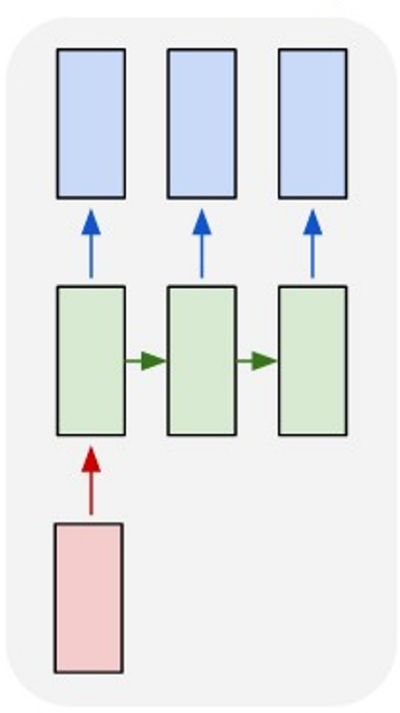
Sequential Learning Problems

one-to-one



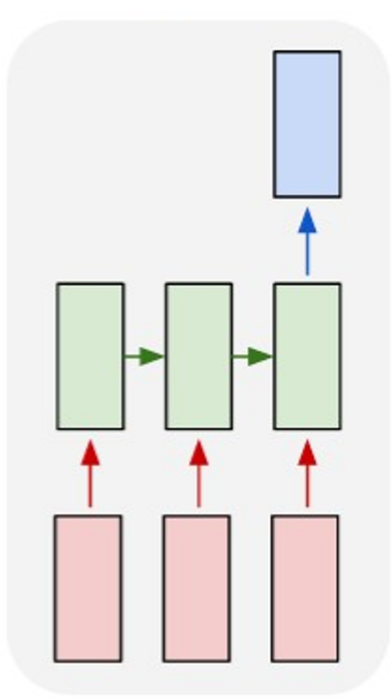
e.g.
image
classification

one-to-many



e.g.
image captioning:
image ->sequence of words

many-to-one



e.g.
video classification:
sequence of images -> label

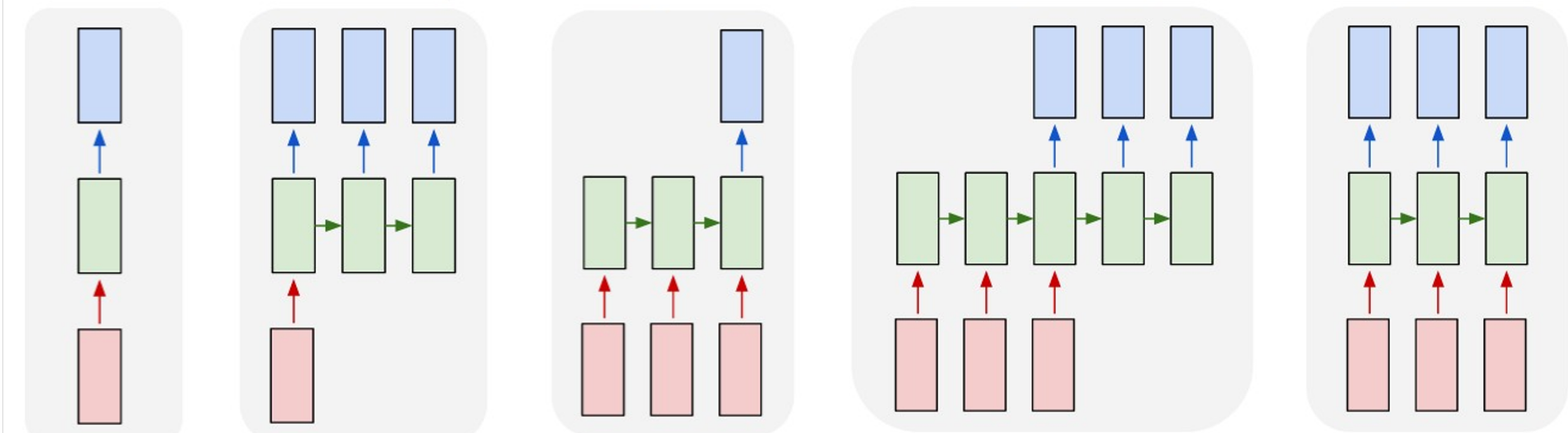
Sequential Learning Problems

one-to-one

one-to-many

many-to-one

many-to-many



e.g.
image
classification

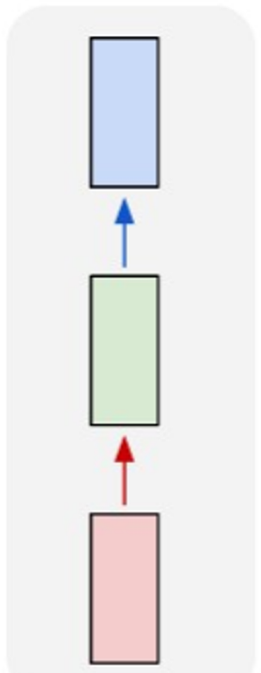
e.g.
image captioning:
image ->sequence of words

e.g.
video classification:
sequence of images -> label

e.g.
machine translation:
sequence of words
->
sequence of words

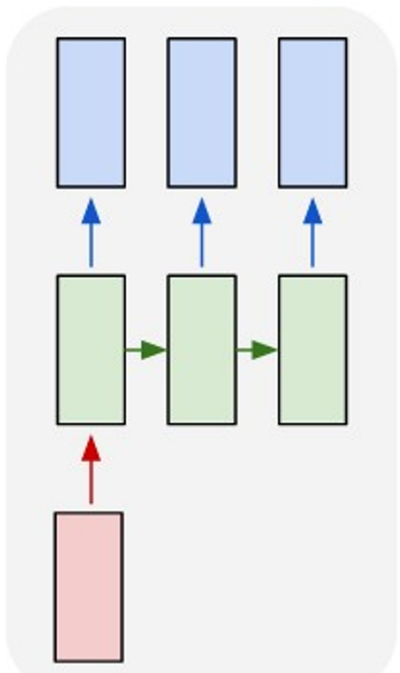
Sequential Learning Problems

one-to-one



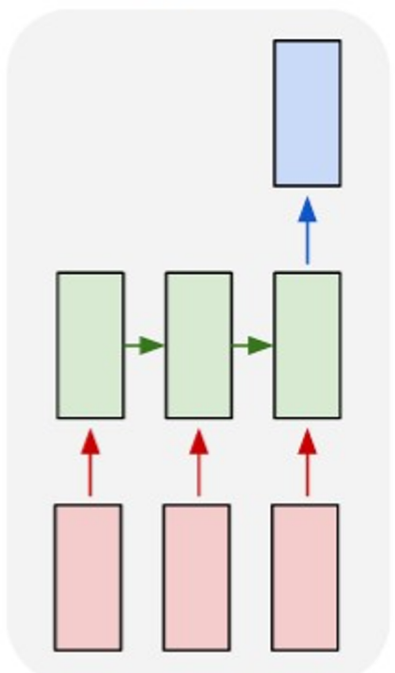
e.g.
image
classification

one-to-many



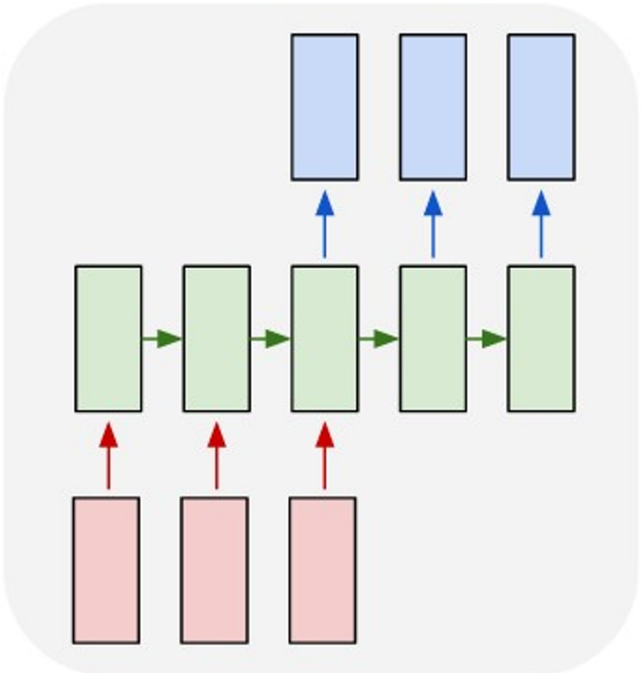
e.g.
image captioning:
image ->sequence of words

many-to-one

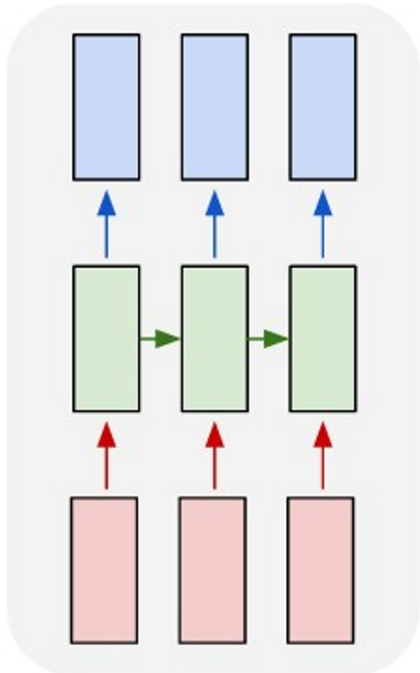


e.g.
video classification:
sequence of images -> label

many-to-many

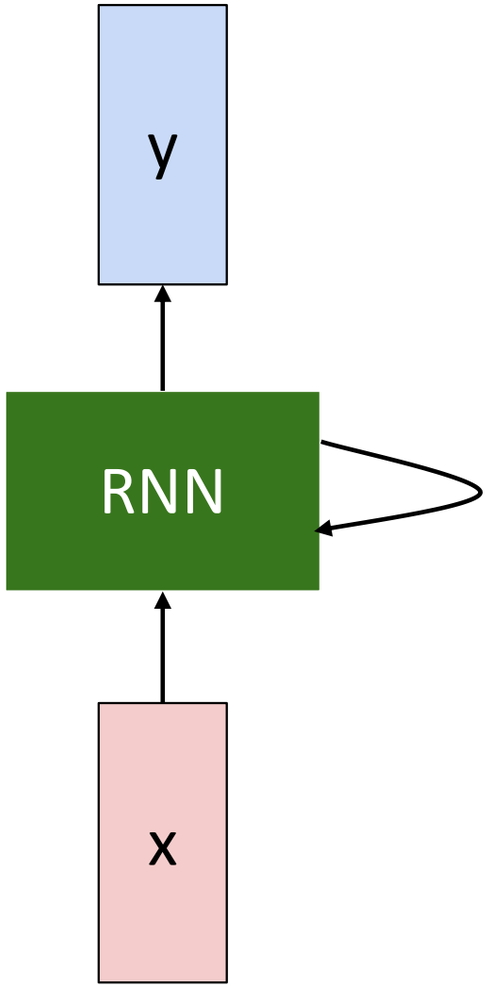


e.g.
machine translation:
sequence of words
->
sequence of words



e.g.
per-frame
video classification
sequence of images
->
sequence of images

Recurrent Neural Networks



We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

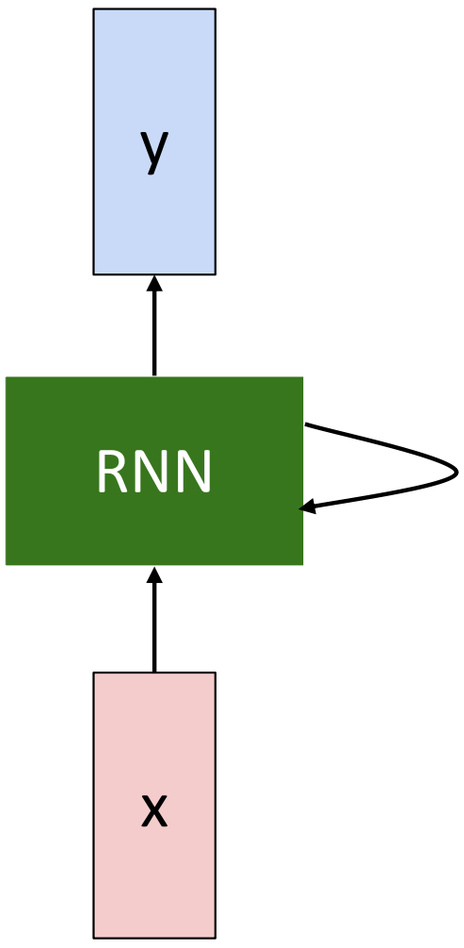
new state

some function with parameters W

old state

input vector at some time step

Vanilla RNN



The state consists of a single *"hidden"* vector **h**:

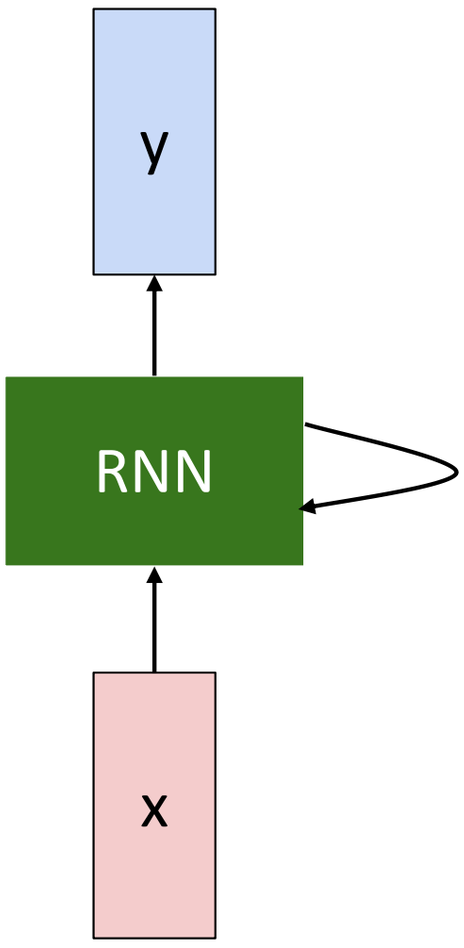
$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

$$y_t = W_{hy}h_t + b_y$$

Sometimes called a "Vanilla RNN" or an "Elman RNN" after Prof. Jeffrey Elman

Vanilla RNN



The state consists of a single *"hidden"* vector **h**:

$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

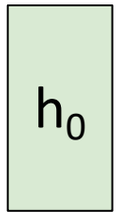
$$y_t = W_{hy}h_t + b_y$$

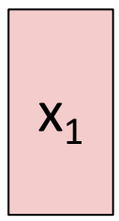
Sometimes called a "Vanilla RNN" or an "Elman RNN" after Prof. Jeffrey Elman

Model weights are not dependent of time.
We have only one model,
not one model for each time!

RNN Computational Graph

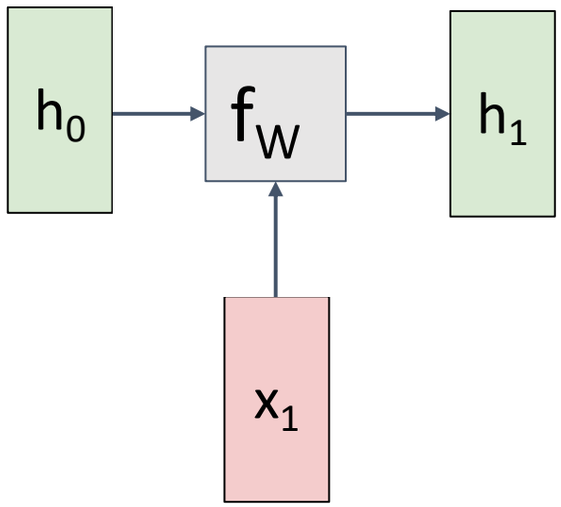
Initial hidden state
Either set to all 0,
Or learn it





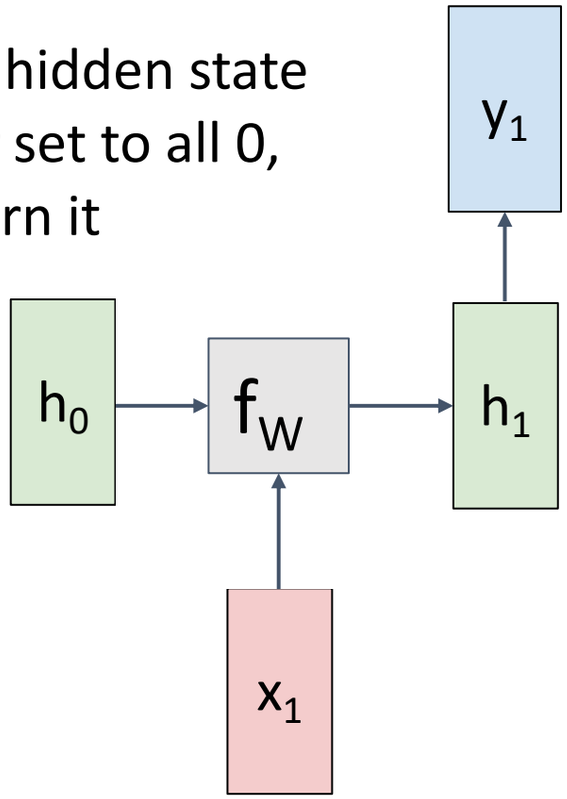
RNN Computational Graph

Initial hidden state
Either set to all 0,
Or learn it



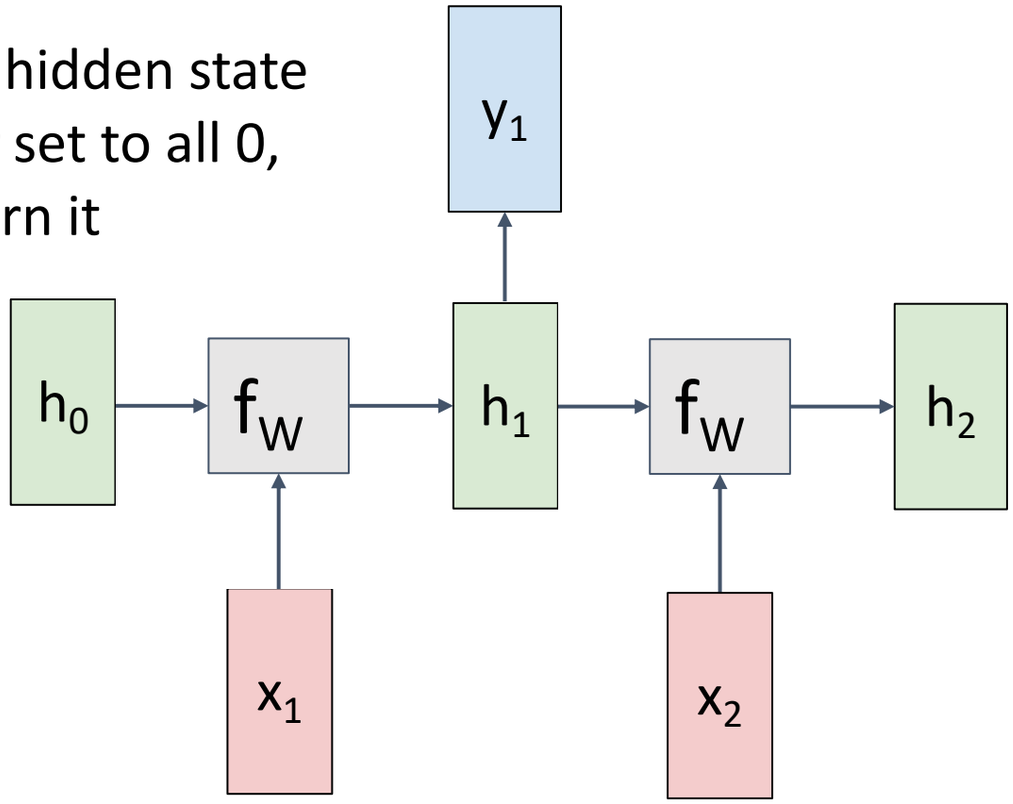
RNN Computational Graph

Initial hidden state
Either set to all 0,
Or learn it



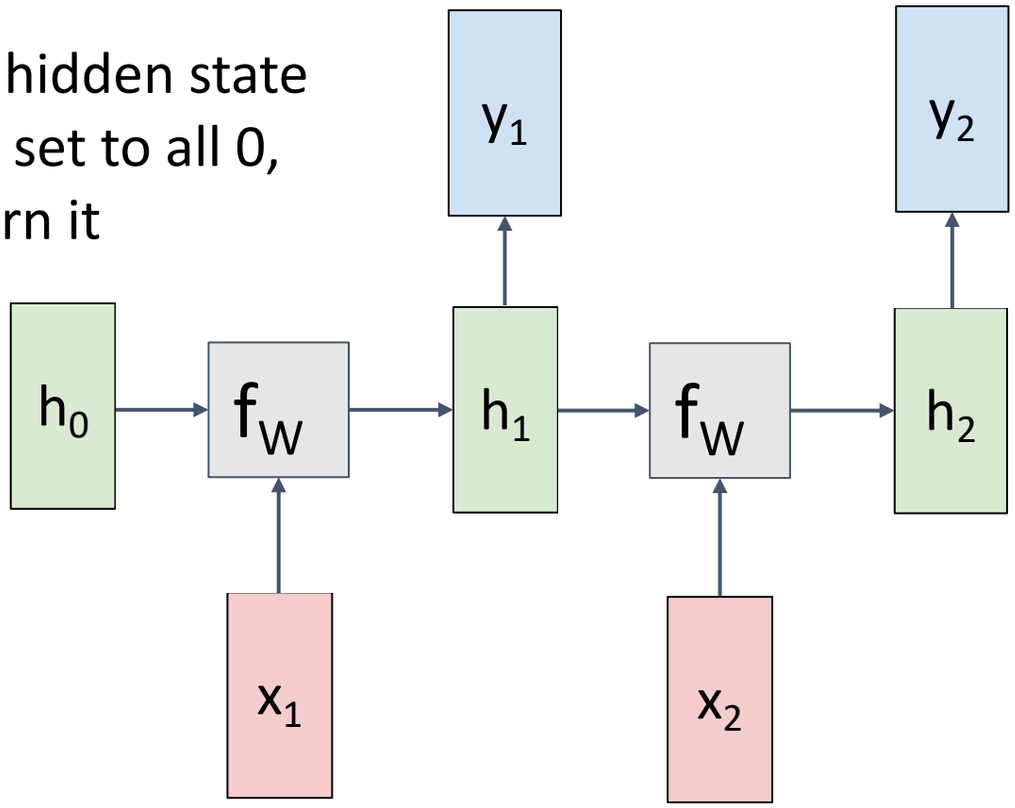
RNN Computational Graph

Initial hidden state
Either set to all 0,
Or learn it

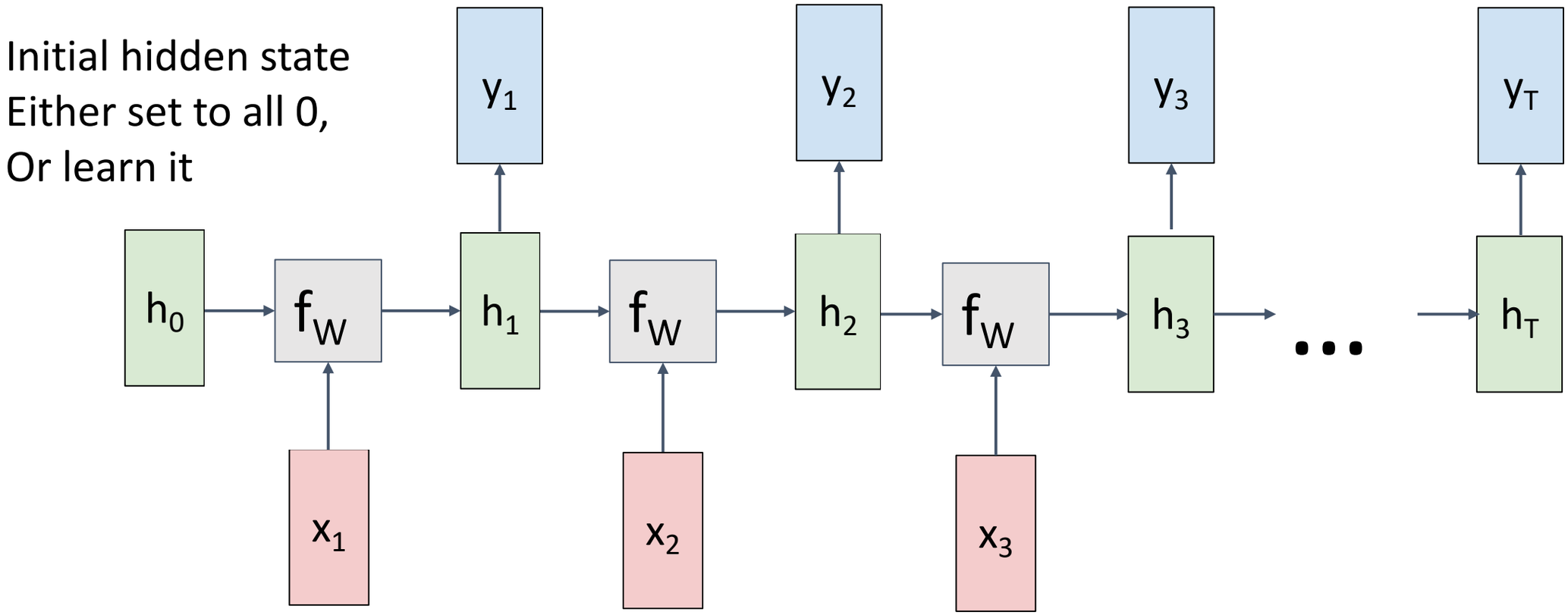


RNN Computational Graph

Initial hidden state
Either set to all 0,
Or learn it

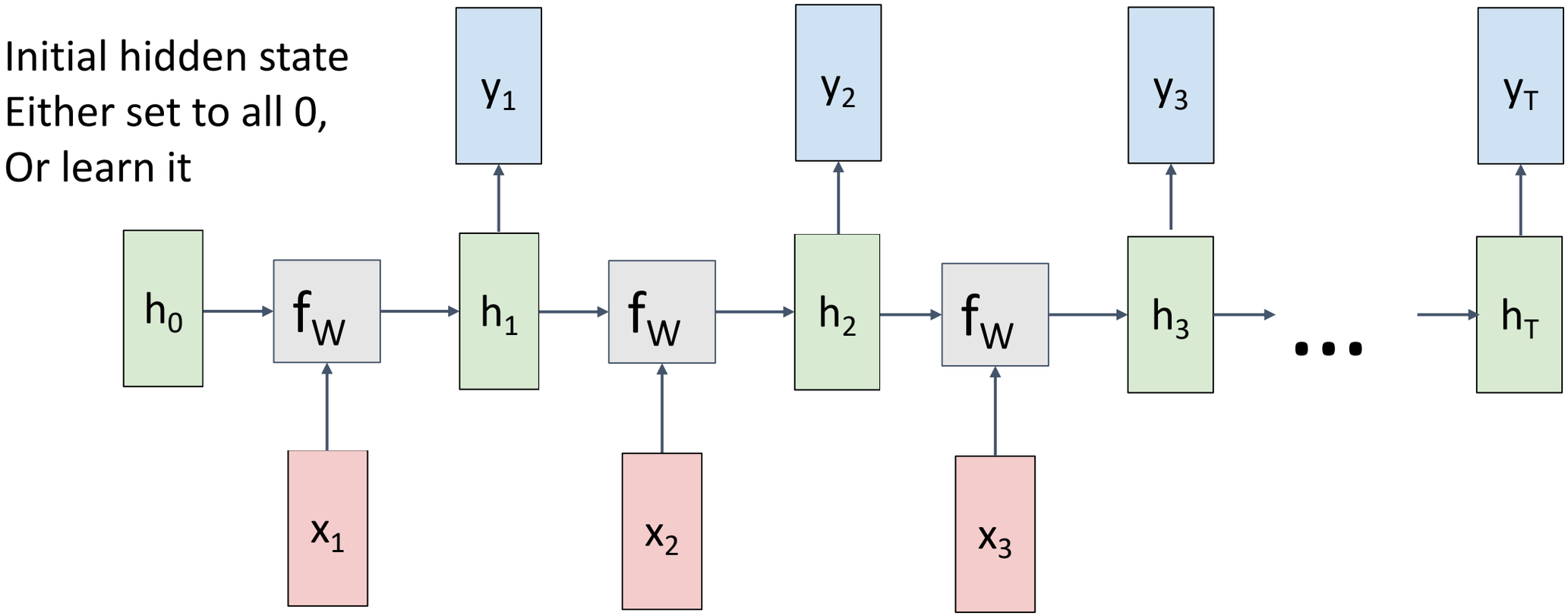


RNN Computational Graph

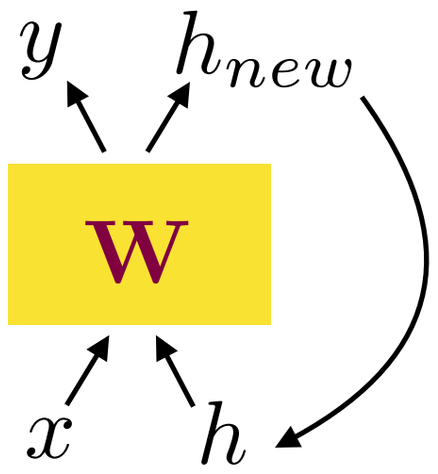


RNN Computational Graph

Initial hidden state
Either set to all 0,
Or learn it

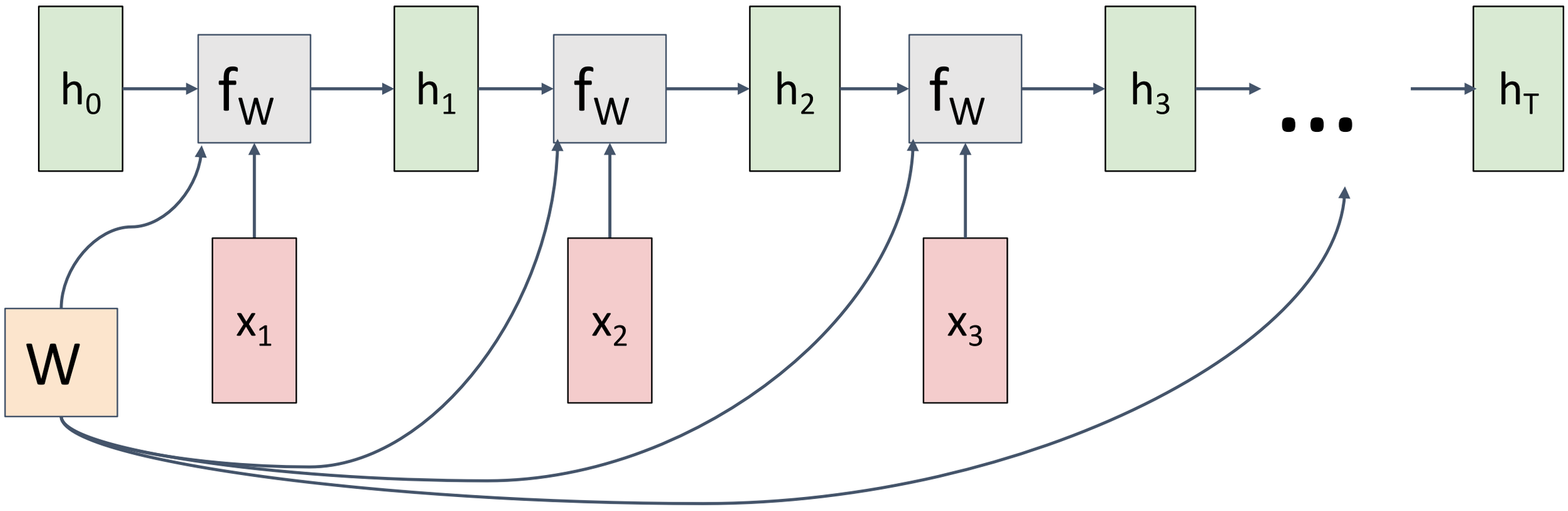


This is only the computational graph.
The model is fixed.

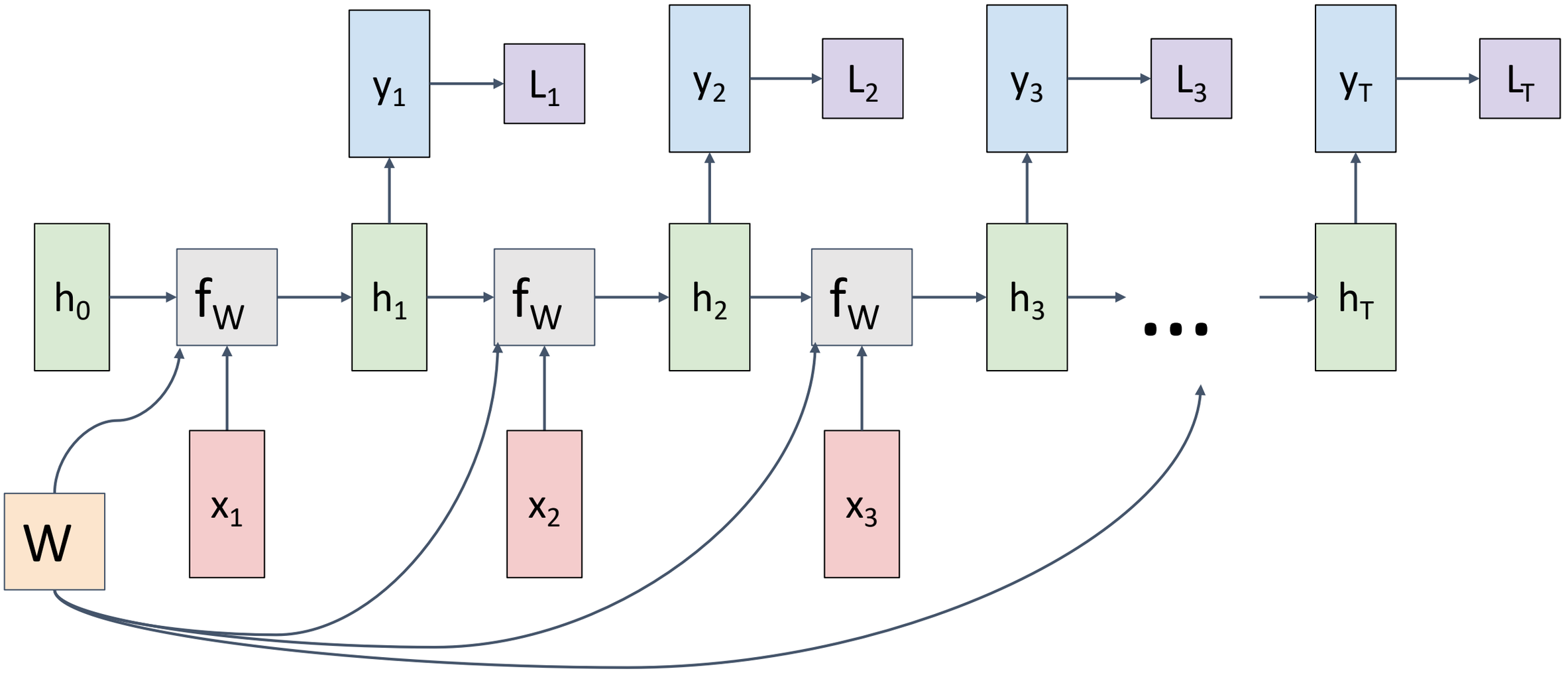


RNN Computational Graph

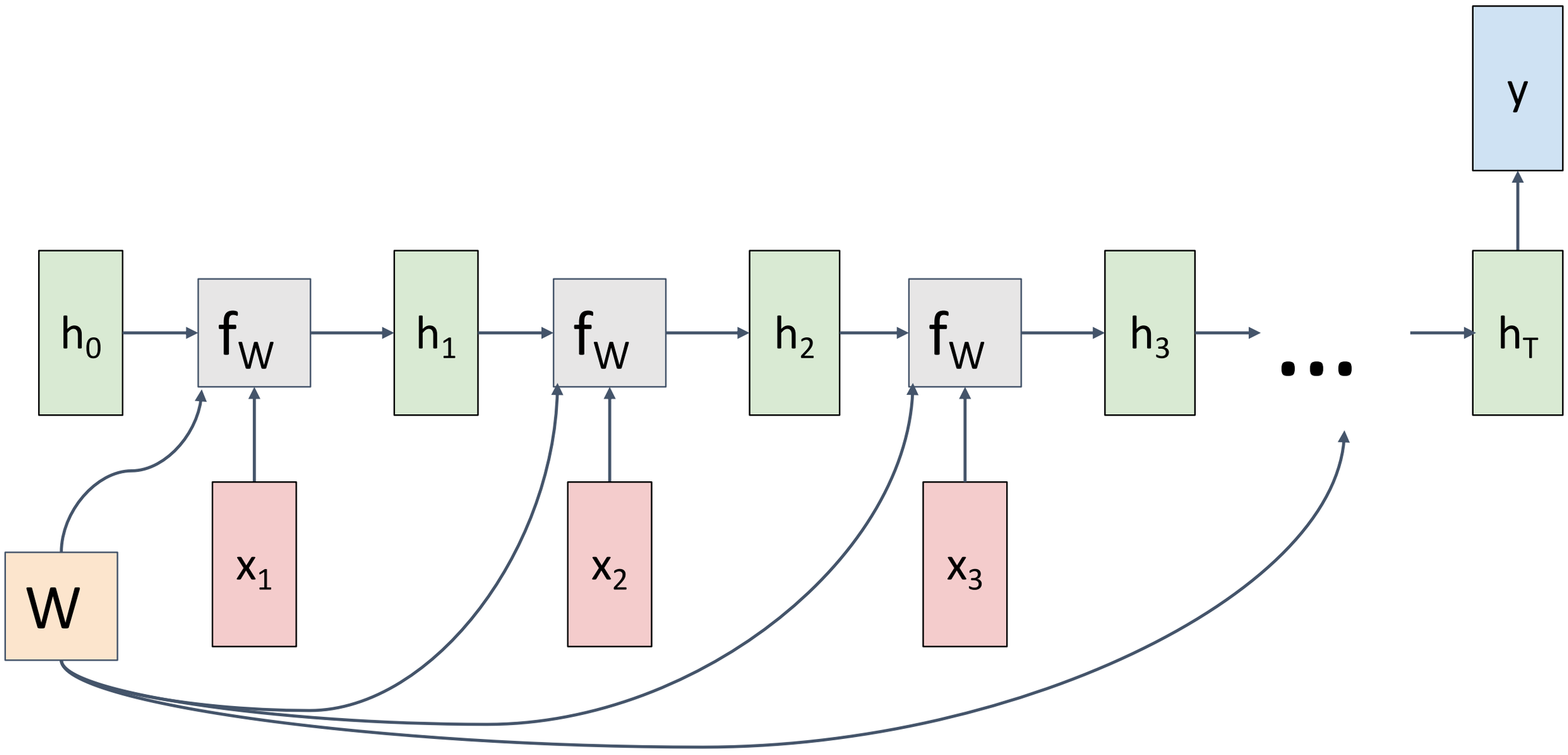
Re-use the same weight matrix at every time-step



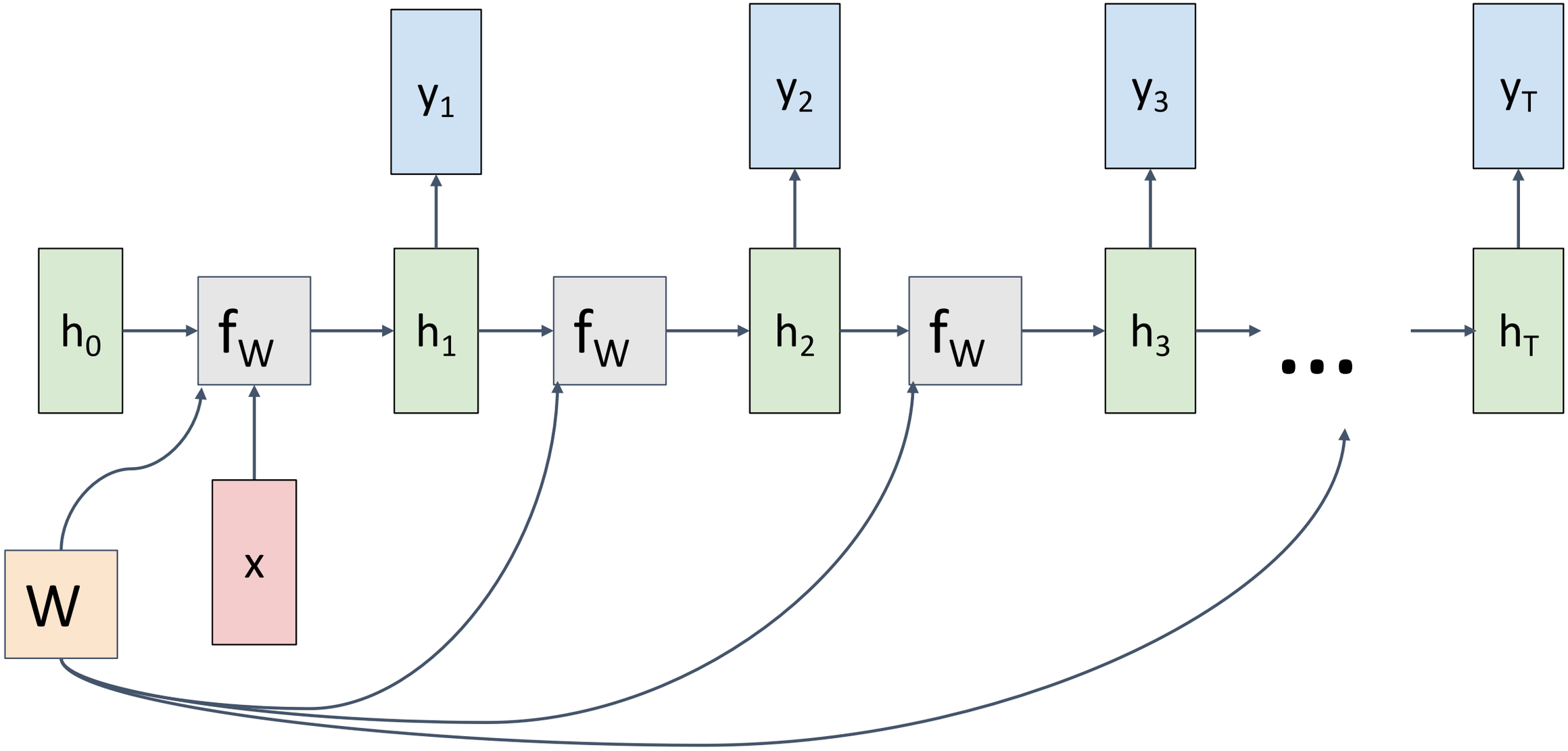
RNN Computational Graph
Many to Many



RNN Computational Graph
Many to One



RNN Computational Graph
One to Many

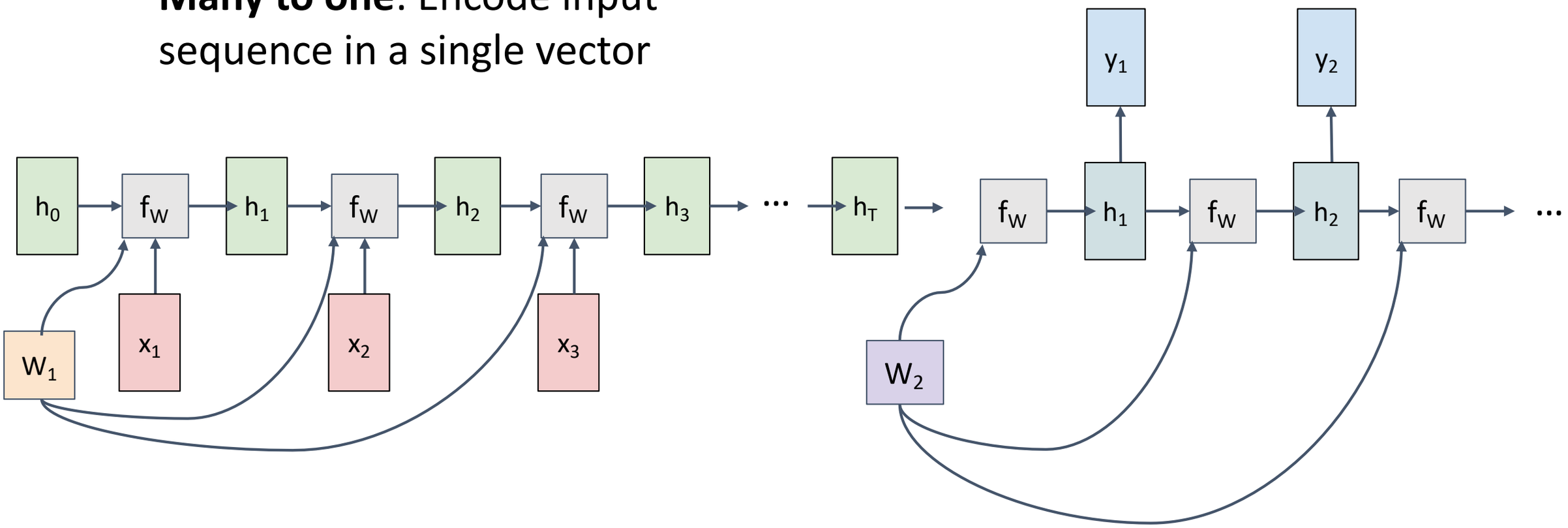


RNN Computational Graph
Sequence to Sequence

Sequence to Sequence (seq2seq)
(Many to one) + (One to many)

Many to one: Encode input sequence in a single vector

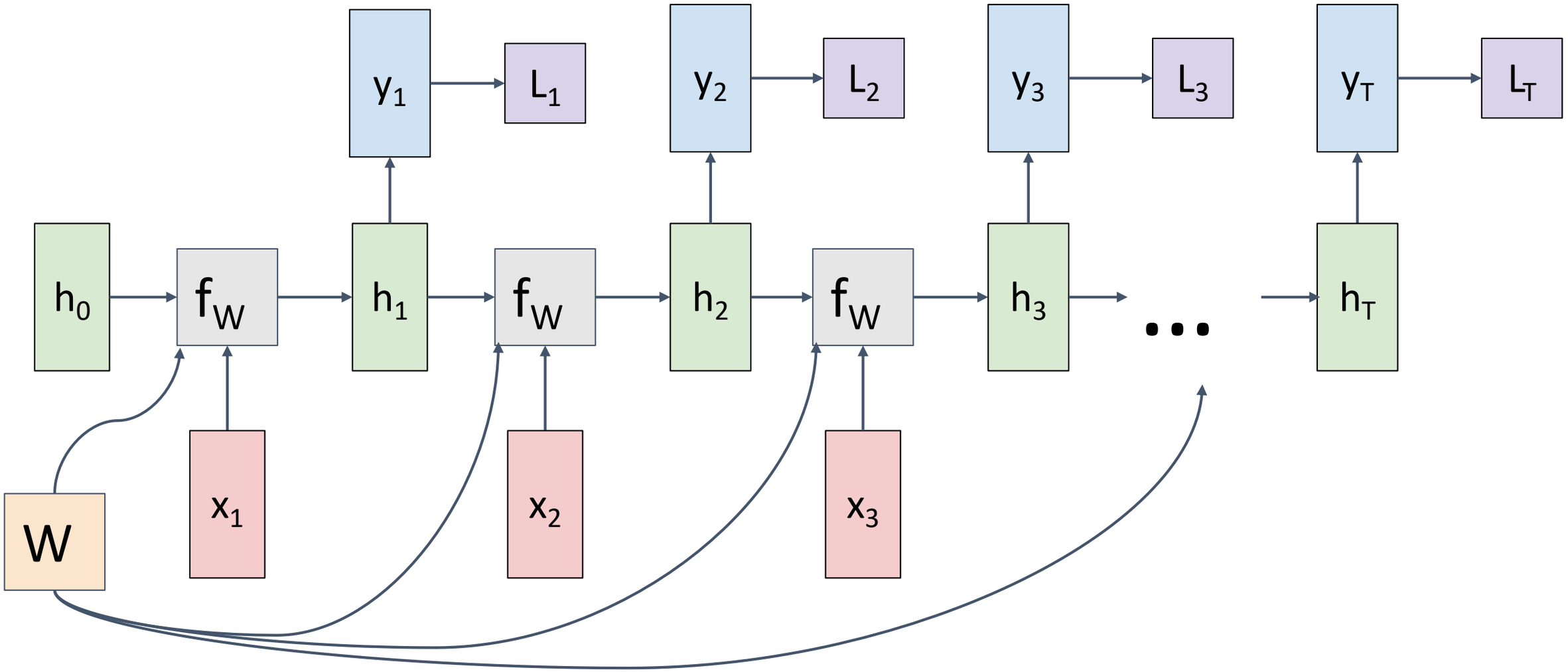
One to many: Produce output sequence from single input vector



Sutskever et al, "Sequence to Sequence Learning with Neural Networks", NIPS 2014

Example: machine translation.

Why not Many to Many?



For sequence to sequence,
the early outputs should also depend on the later inputs.

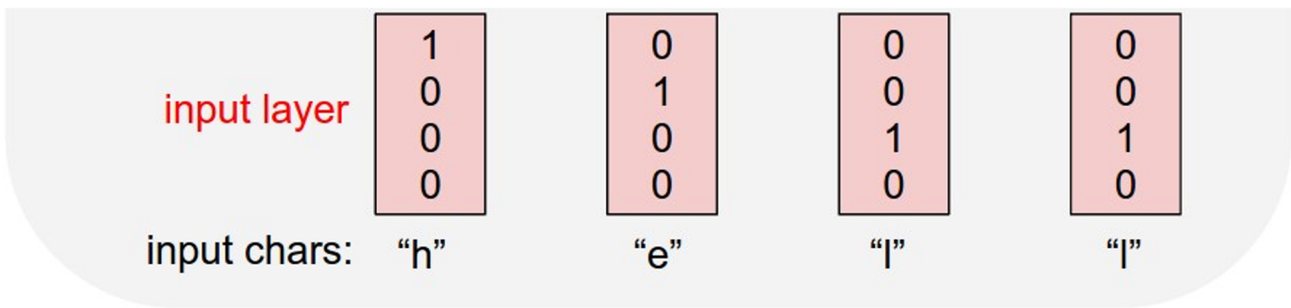
Language Modeling with RNN

Given characters 1, 2, …, t-1,
model predicts character t

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

Training sequence: ”hello”

Vocabulary: [h, e, l, o]



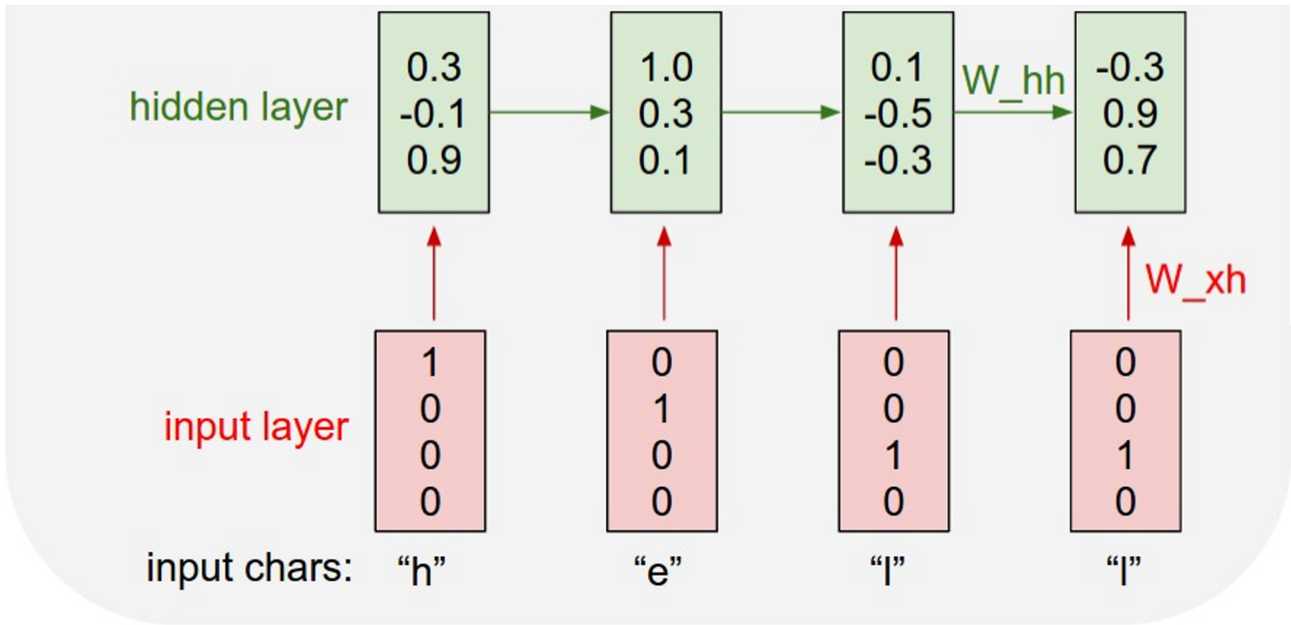
Language Modeling with RNN

Given characters 1, 2, …, t-1,
model predicts character t

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

Training sequence: ”hello”

Vocabulary: [h, e, l, o]



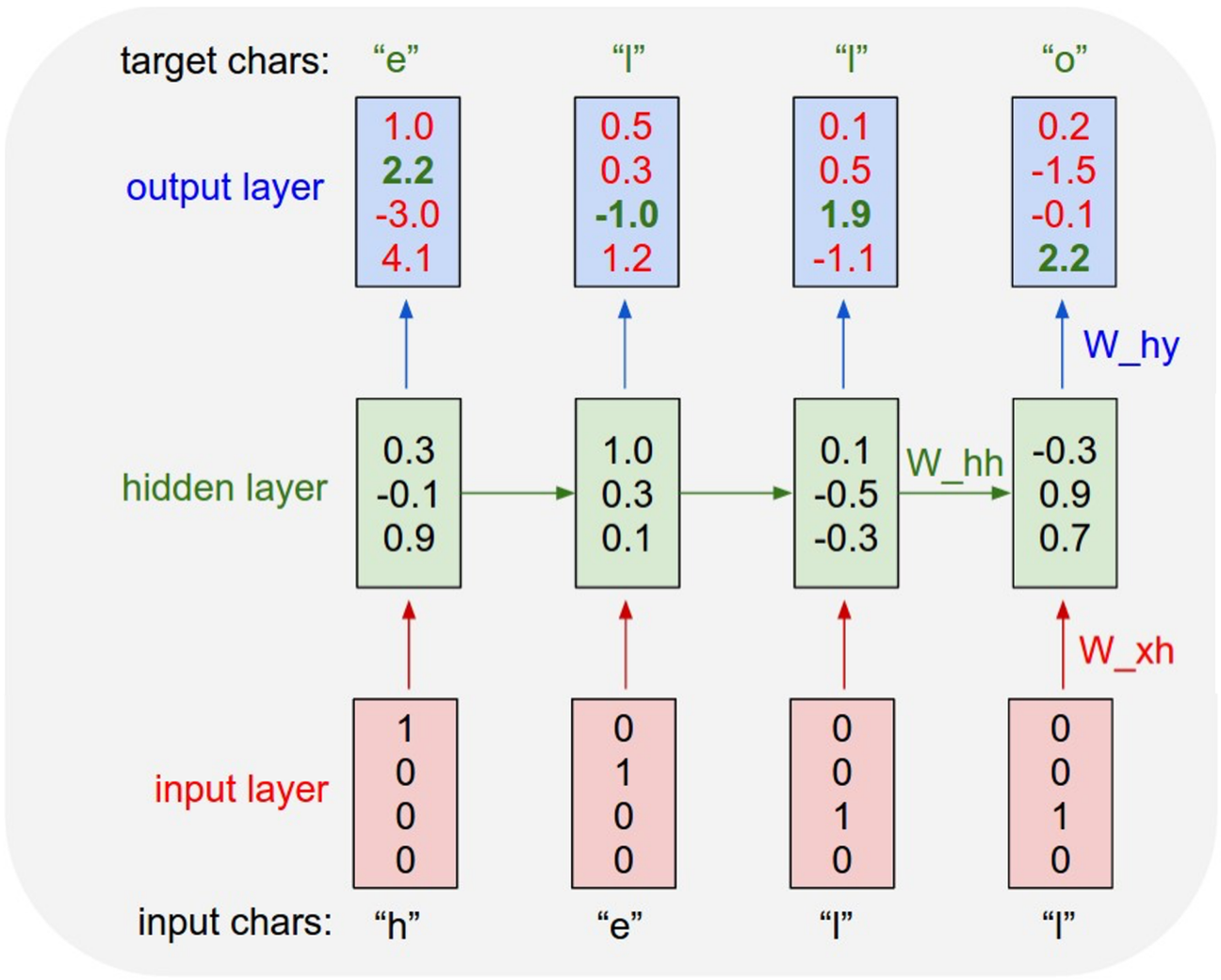
Language Modeling with RNN

Given characters 1, 2, …, t-1,
model predicts character t

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

Training sequence: ”hello”

Vocabulary: [h, e, l, o]

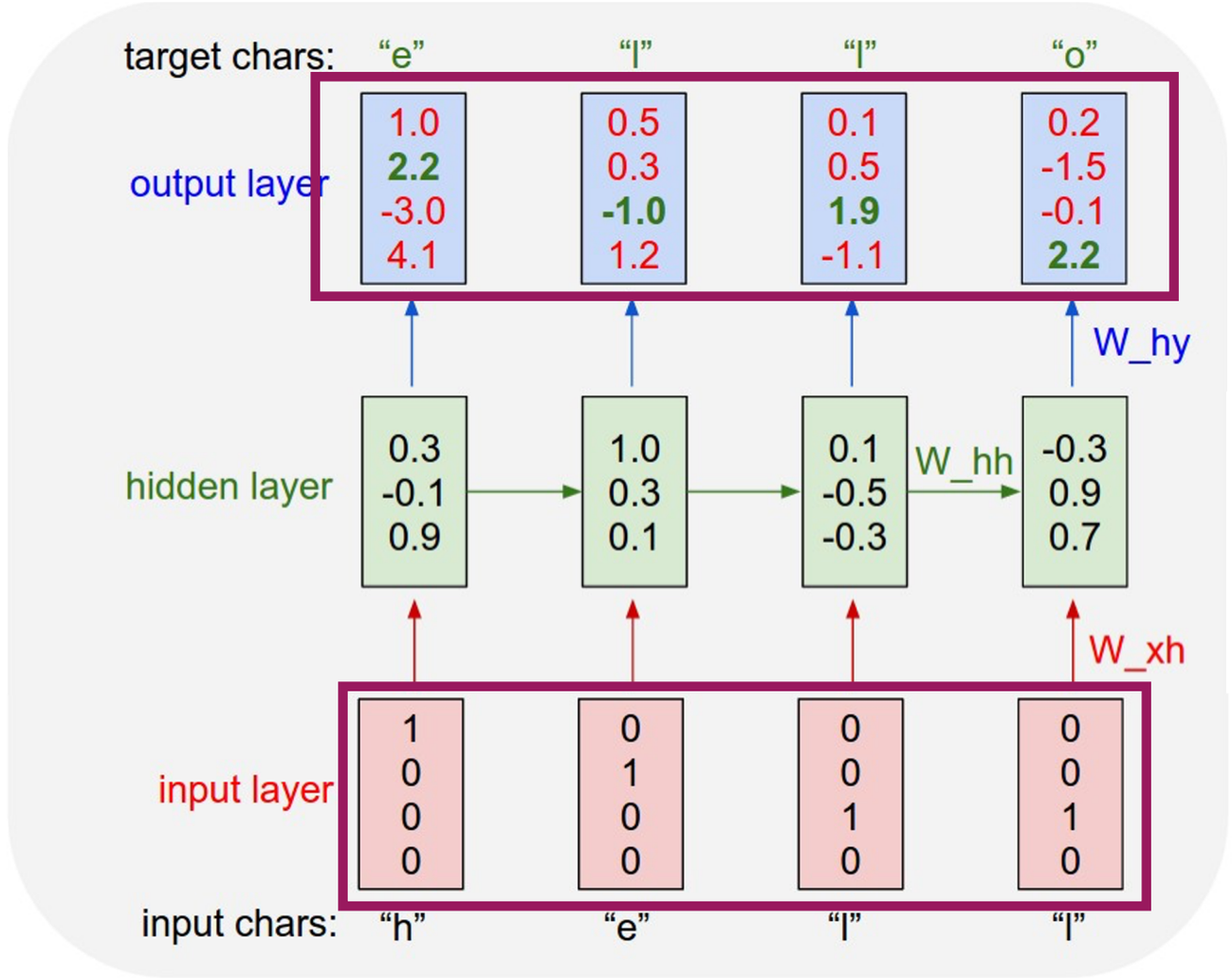


Language Modeling with RNN

Given characters 1, 2, ..., t-1, model predicts character t

Input: one-hot encoding of vocabulary.
Output: (unnormalized) softmax output of vocabulary.
Training: cross-entropy loss.

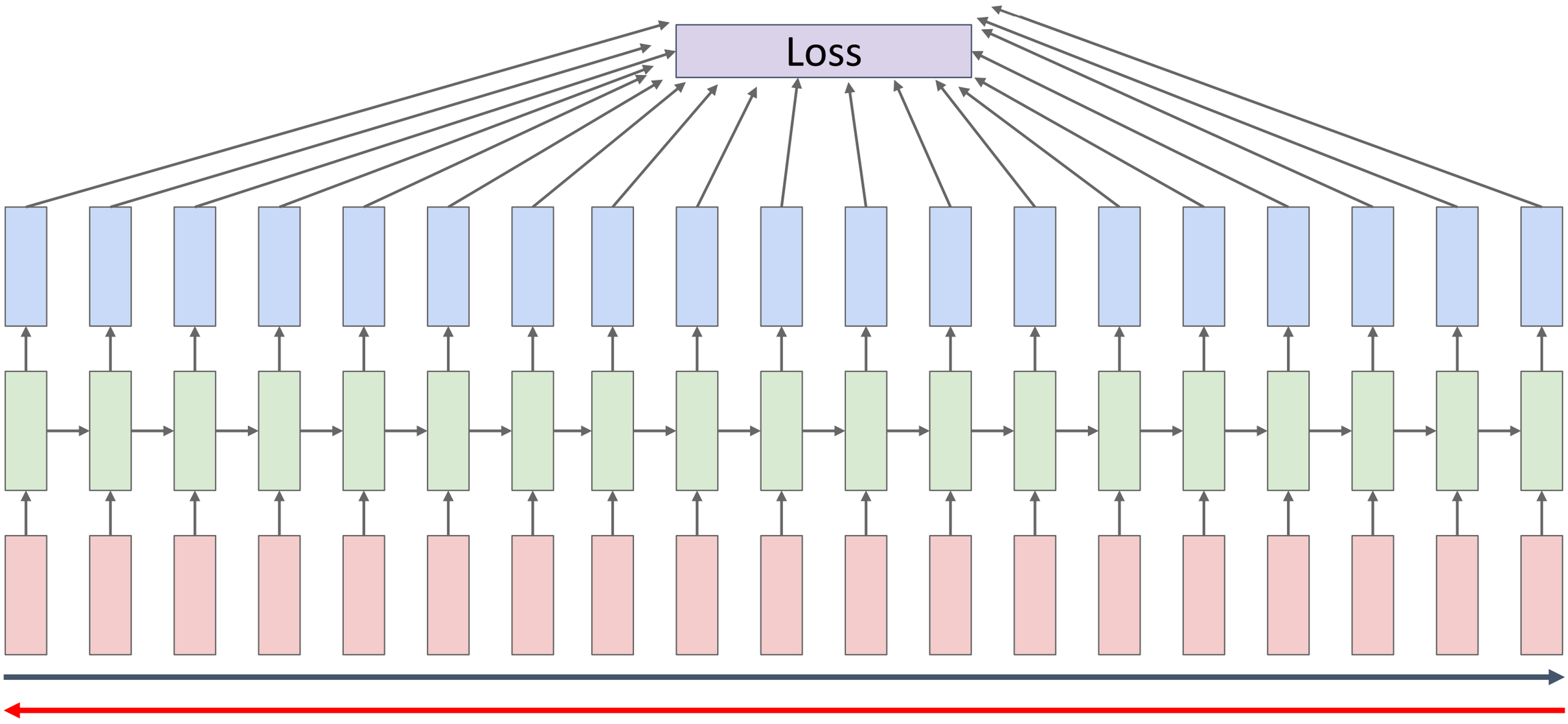
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$



Training sequence: "hello"

Vocabulary: [h, e, l, o]

Backpropagation Through Time



The gradient should be calculated through both model and time.

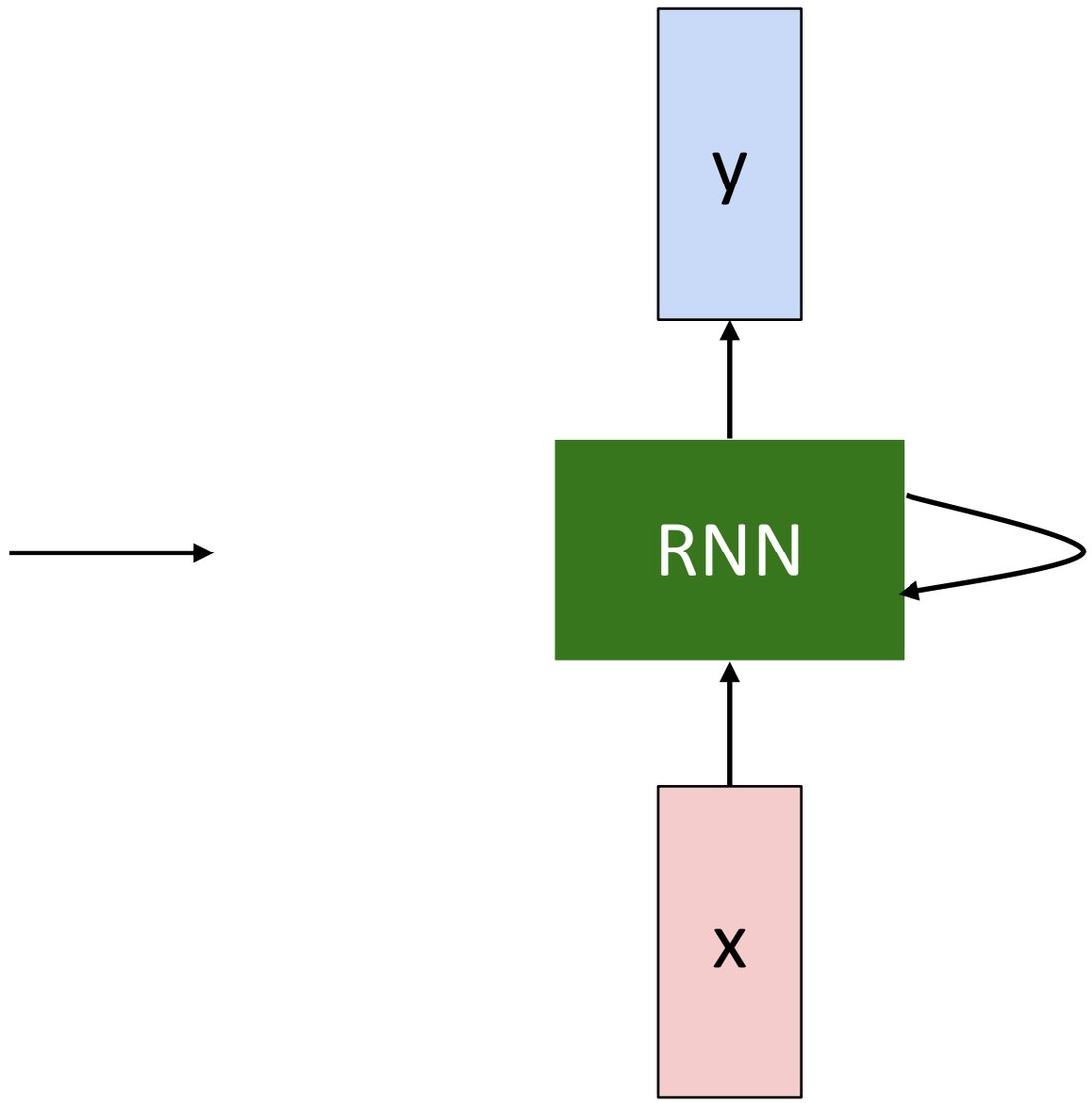
Text Generation with RNN

THE SONNETS

by William Shakespeare

From fairest creatures we desire increase,
That thereby beauty's rose might never die,
But as the riper should by time decease,
His tender heir might bear his memory:
But thou, contracted to thine own bright eyes,
Feed'st thy light's flame with self-substantial fuel,
Making a famine where abundance lies,
Thyself thy foe, to thy sweet self too cruel:
Thou that art now the world's fresh ornament,
And only herald to the gaudy spring,
Within thine own bud buriest thy content,
And tender churl mak'st waste in niggarding:
    Pity the world, or else this glutton be,
    To eat the world's due, by the grave and thee.

When forty winters shall besiege thy brow,
And dig deep trenches in thy beauty's field,
Thy youth's proud livery so gazed on now,
Will be a tatter'd weed of small worth held:
Then being asked, where all thy beauty lies,
Where all the treasure of thy lusty days;
To say, within thine own deep sunken eyes,
Were an all-eating shame, and thriftless praise.
How much more praise deserv'd thy beauty's use,
If thou couldst answer 'This fair child of mine
Shall sum my count, and make my old excuse,'
Proving his beauty by succession thine!
    This were to be new made when thou art old,
    And see thy blood warm when thou feel'st it cold.



Text Generation with RNN

at first:

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

Text Generation with RNN

at first:

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

train more

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

Text Generation with RNN

at first:

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

train more

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

train more

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

Text Generation with RNN

at first:

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

train more

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

train more

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

train more

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

Text Generation with RNN

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```

```
VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.
```

Code Generation

```
static void do_command(struct seq_file *m, void *v)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
    cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
  else
    seq = 1;
  for (i = 0; i < 16; i++) {
    if (k & (1 << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
        ((count & 0x00000000fffffff8) & 0x000000f) << 8;
    if (count == 0)
      sub(pid, ppc_md.kexec_handle, 0x20000000);
    pipe_set_bytes(i, 0);
  }
  /* Free our user pages pointer to place camera if all dash */
  subsystem_info = &of_changes[PAGE_SIZE];
  rek_controls(offset, idx, &soffset);
  /* Now we want to deliberately put it to device */
  control_check_polarity(&context, val, 0);
  for (i = 0; i < COUNTER; i++)
    seq_puts(s, "policy ");
}
```

Generated C code

Image Captioning

Image Captioning: Example Results

Captions generated using neuraltalk2
All images are CC0 Public domain: cat suitcase, cat tree, dog, bear, surfers, tennis, giraffe, motorcycle

A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

Two people walking on the beach with surfboards

A tennis player in action on the court

Two giraffes standing in a grassy field

A man riding a dirt bike on a dirt track

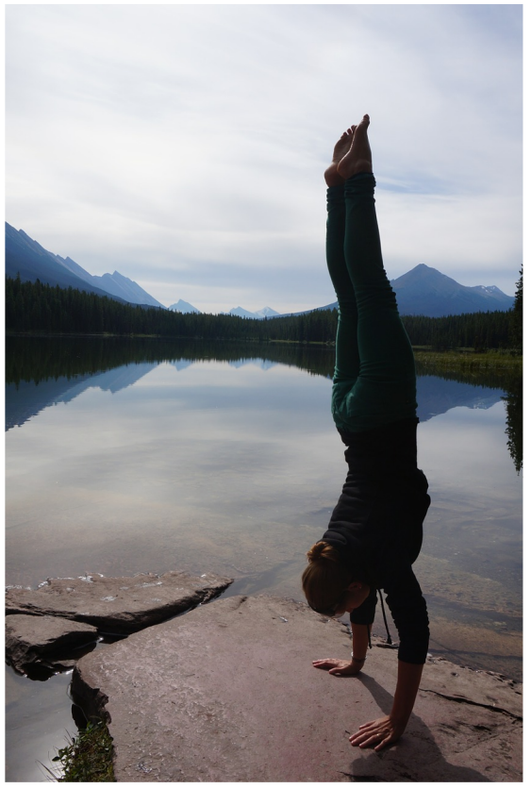
Image Captioning

Captions generated using neuraltalk2
All images are CC0 Public domain: fur coat, handstand, spider web, baseball

Image Captioning: Failure Cases

A woman is holding a cat in her hand

A person holding a computer mouse on a desk

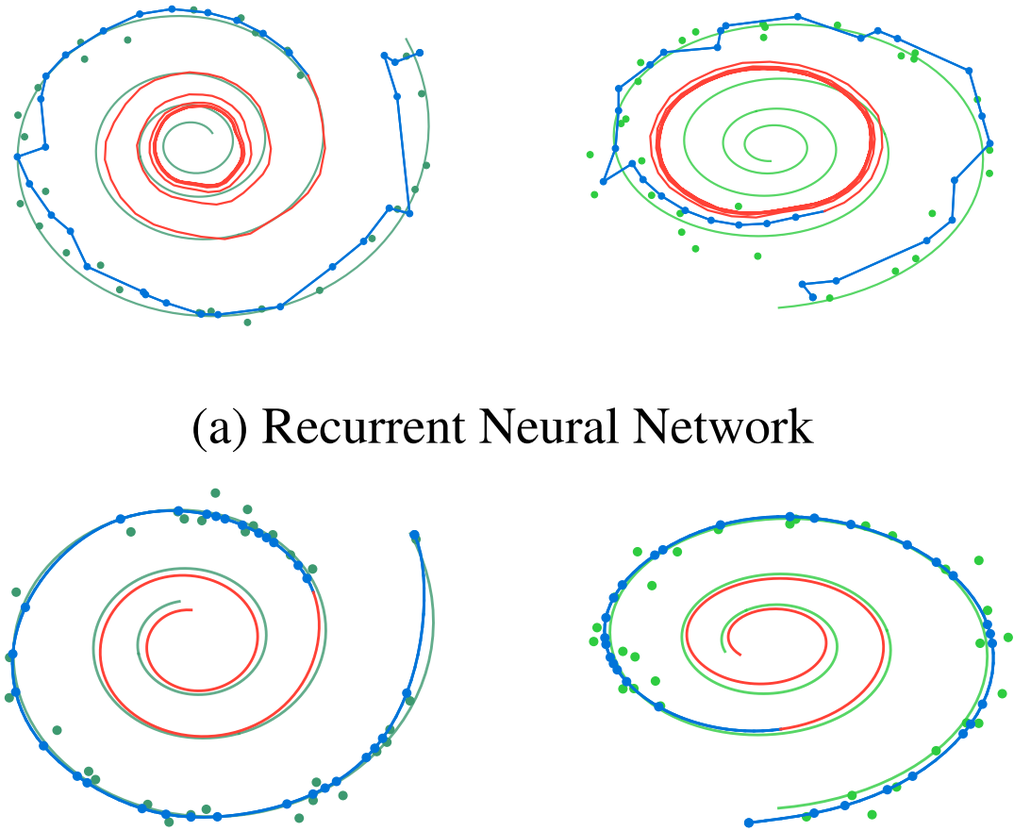


A woman standing on a beach holding a surfboard

A bird is perched on a tree branch

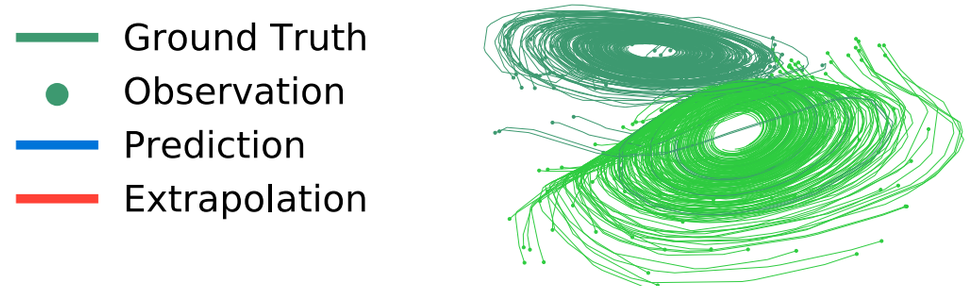
A man in a baseball uniform throwing a ball

RNNs are ODEs



(a) Recurrent Neural Network

(b) Latent Neural Ordinary Differential Equation



RNNs actually represent discrete dynamic systems, which are ordinary differential equations w.r.t. time.

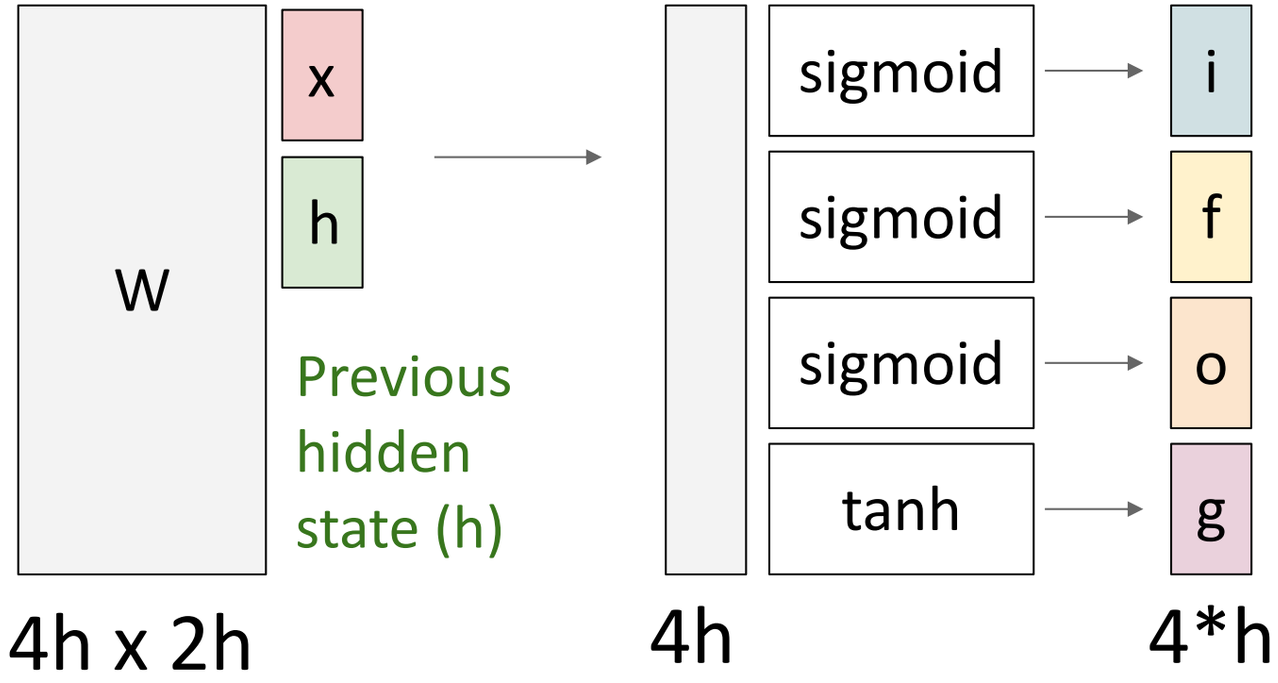
Not all ODEs have stable solutions.
RNNs may suffer from instability issue.

Neural Ordinary Differential Equations, NeurIPS'18.

Practical RNN Models

Long Short Term Memory (LSTM)

i: Input gate, whether to write to cell
f: Forget gate, Whether to erase cell
o: Output gate, How much to reveal cell
g: Gate gate (?), How much to write to cell



$$\begin{pmatrix} i_t \\ f_t \\ o_t \\ g_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_h \right)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

$$h_t = o_t \odot \tanh(c_t)$$

Practical RNN Models

Other RNN Variants

Gated Recurrent Unit (GRU)
Cho et al "Learning phrase representations using RNN encoder-decoder for statistical machine translation", 2014

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)$$
$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)$$
$$\tilde{h}_t = \tanh(W_{xh}x_t + W_{hh}(r_T \odot h_{t-1}) + b_h)$$
$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t$$

10,000 architectures with evolutionary search:
Jozefowicz et al, "An empirical exploration of recurrent network architectures", ICML 2015

MUT1:

$$\begin{aligned} z &= \text{sigm}(W_{\text{xz}}x_t + b_{\text{z}}) \\ r &= \text{sigm}(W_{\text{xr}}x_t + W_{\text{hr}}h_t + b_{\text{r}}) \\ h_{t+1} &= \tanh(W_{\text{hh}}(r \odot h_t) + \tanh(x_t) + b_{\text{h}}) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

MUT2:

$$\begin{aligned} z &= \text{sigm}(W_{\text{xz}}x_t + W_{\text{hz}}h_t + b_{\text{z}}) \\ r &= \text{sigm}(x_t + W_{\text{hr}}h_t + b_{\text{r}}) \\ h_{t+1} &= \tanh(W_{\text{hh}}(r \odot h_t) + W_{xh}x_t + b_{\text{h}}) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

MUT3:

$$\begin{aligned} z &= \text{sigm}(W_{\text{xz}}x_t + W_{\text{hz}}\tanh(h_t) + b_{\text{z}}) \\ r &= \text{sigm}(W_{\text{xr}}x_t + W_{\text{hr}}h_t + b_{\text{r}}) \\ h_{t+1} &= \tanh(W_{\text{hh}}(r \odot h_t) + W_{xh}x_t + b_{\text{h}}) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

Machine Learning: V

- NN to process sequential data
  - Recurrent NN
- Generative NN models
  - Autoencoders
  - GANs
- Take-home messages

Unsupervised Learning

Supervised Learning

Data: (x, y)

x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

Unsupervised Learning

Supervised Learning

Data: (x, y)

x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

Unsupervised Learning

Data: x

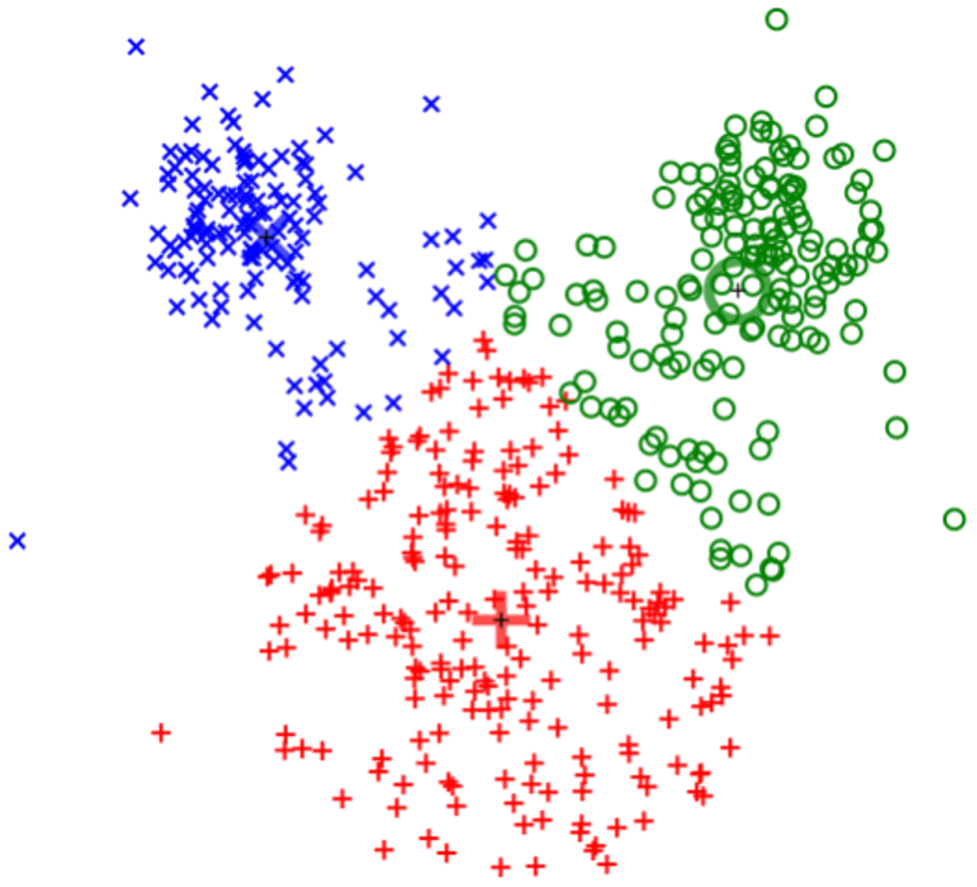
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

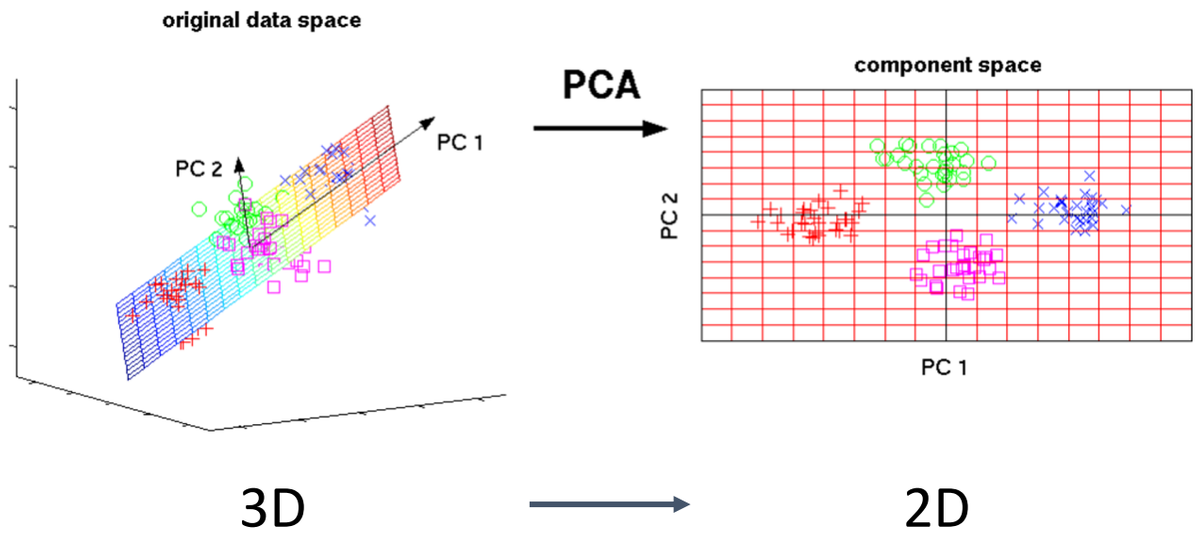
Unsupervised Learning

Clustering (e.g. K-Means)



This image is CC0 public domain

Dimensionality Reduction (e.g. Principal Components Analysis)

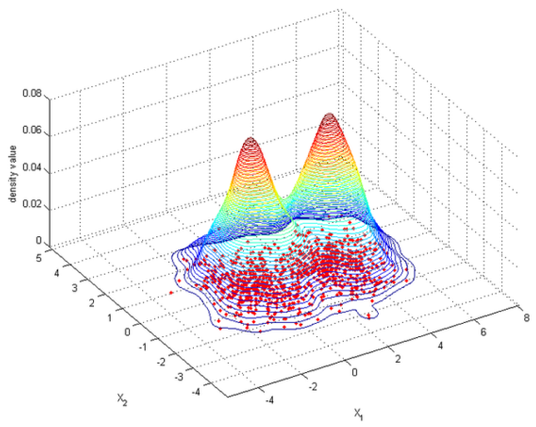


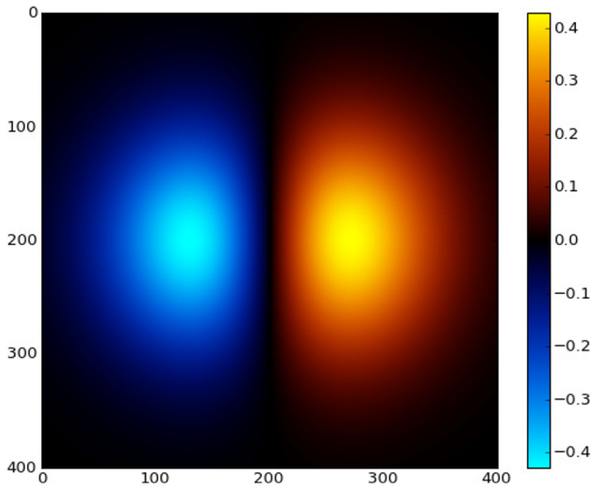
This image from Matthias Scholz is CC0 public domain

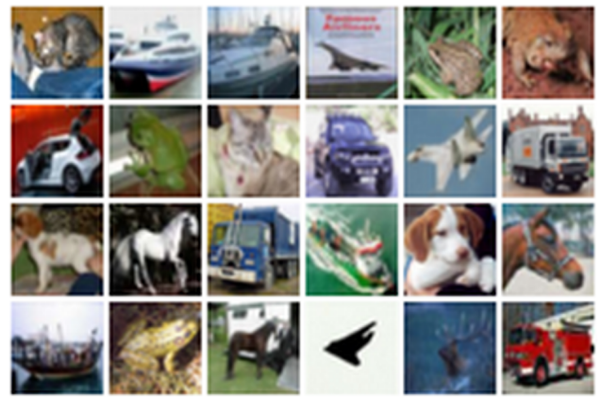
Traditionally important topics.
Skipped due to time limitation.

Unsupervised Learning

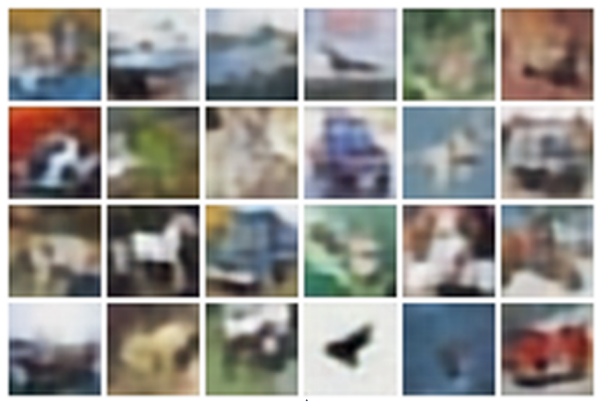
Density Estimation







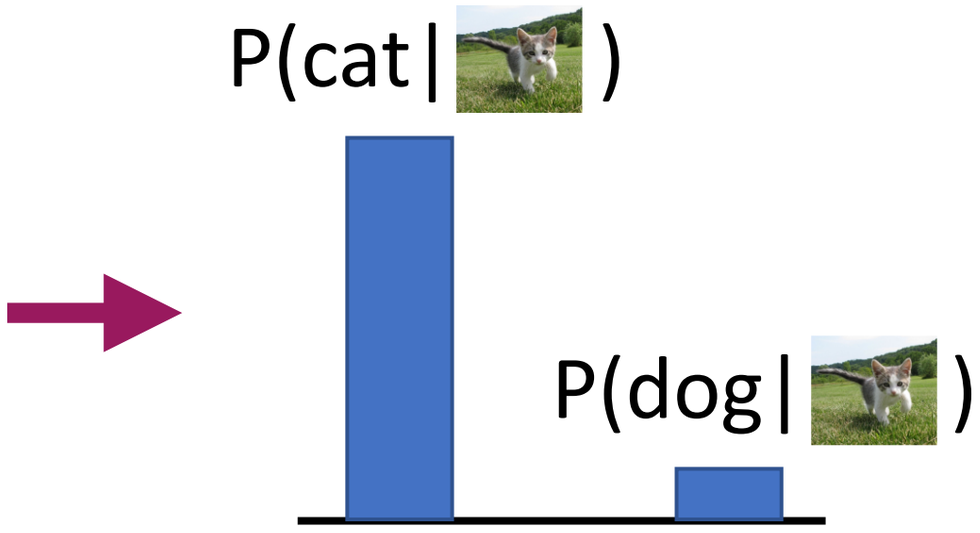
true



estimated

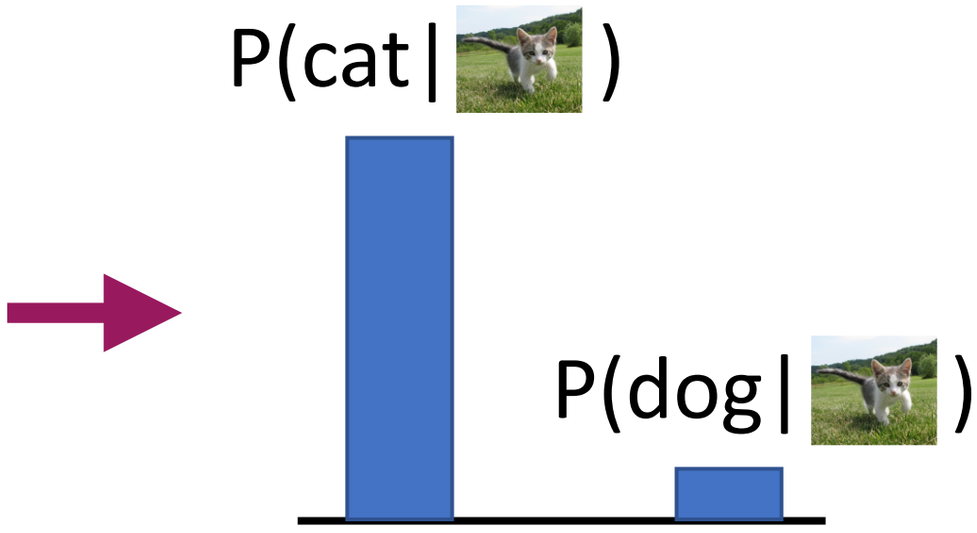
Estimate $p(x)$: the distribution of inputs.

Generative Models

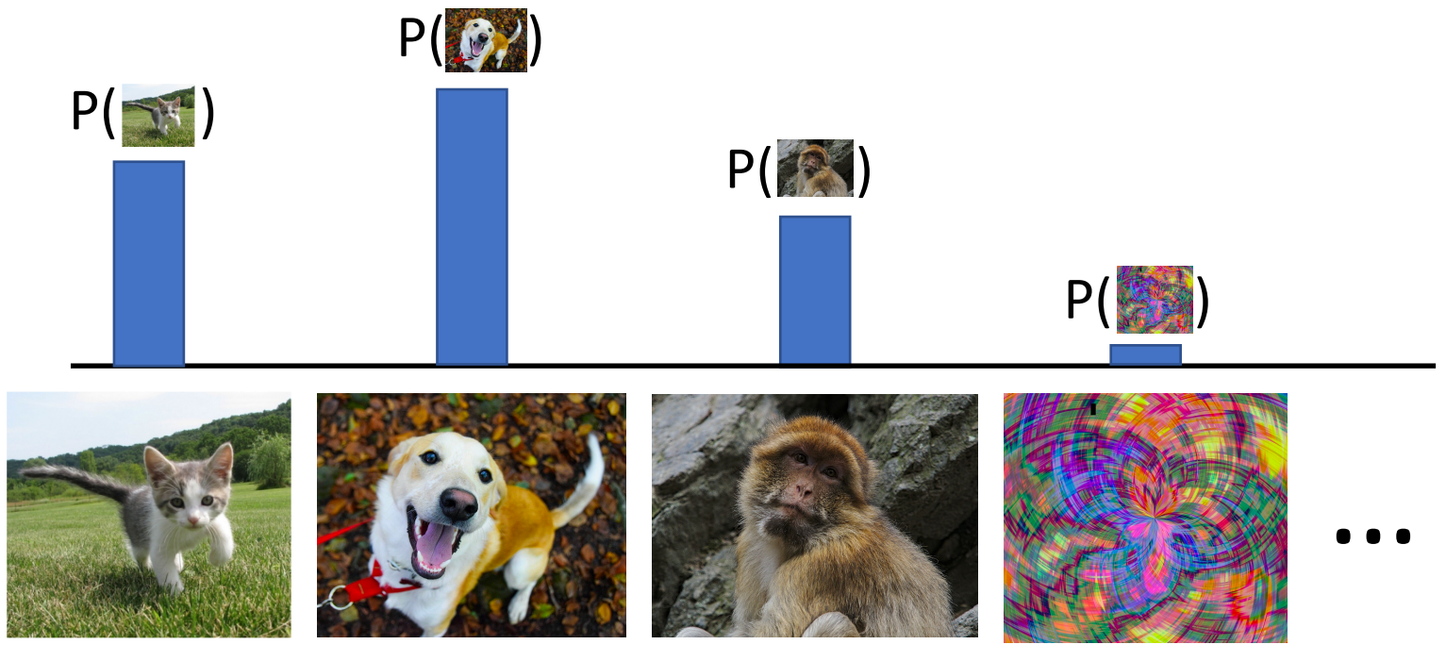


Discriminative Model: Learn a probability distribution p(y|x)

Generative Models



Discriminative Model: Learn a probability distribution p(y|x)



Generative model: All possible images compete with each other for probability mass

Generative Model: Learn a probability distribution p(x)

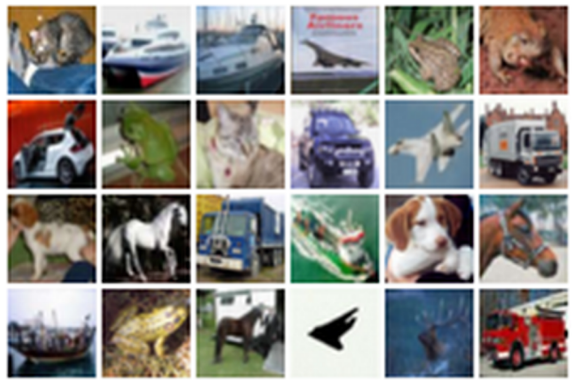
Cat image is CC0 public domain
Dog image is CC0 Public Domain
Monkey image is CC0 Public Domain
Abstract image is free to use under the Pixabay license

Machine Learning: V

- NN to process sequential data
  - Recurrent NN
  - LSTM & GRU
- Generative NN models
  - Autoencoders
  - GANs
- Take-home messages

Autoencoder

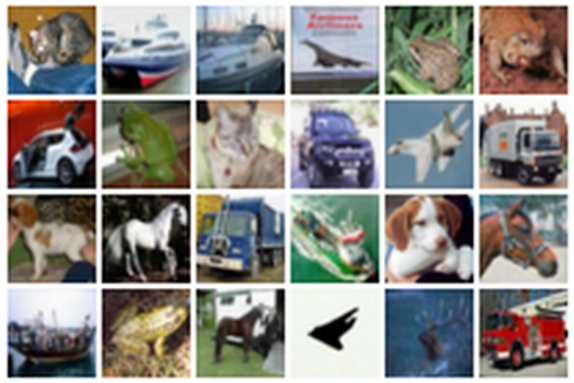
- An autoencoder consists of both an encoder and a decoder:



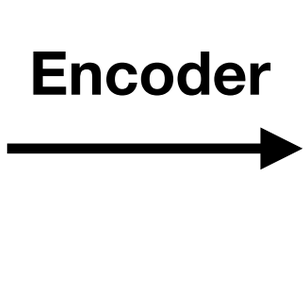
Input Data

Autoencoder

- An autoencoder consists of both an encoder and a decoder:
  - Encoder: transform input x into latent representation z



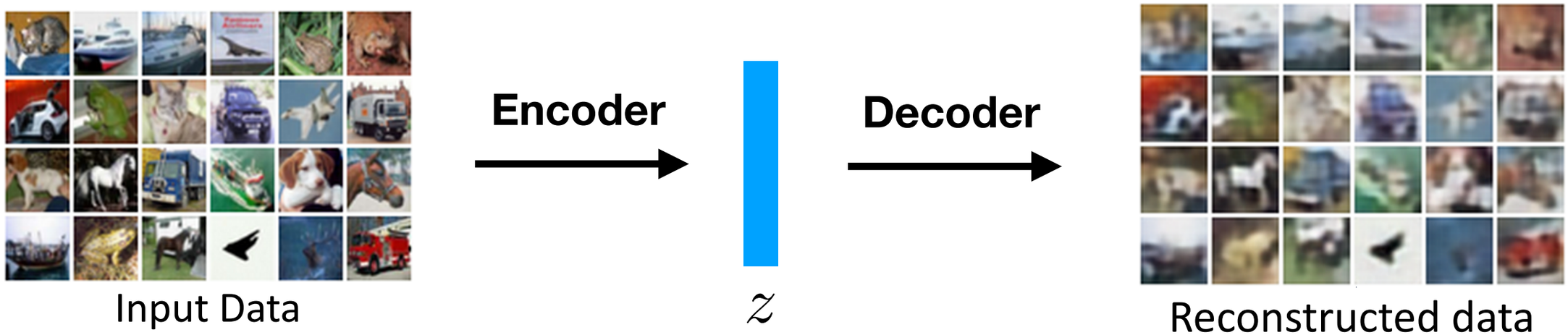
Input Data



z

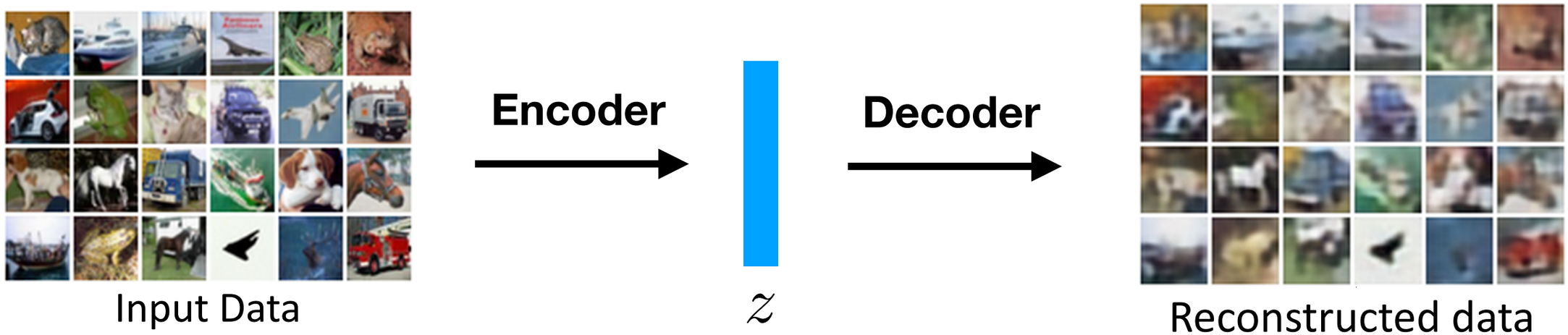
Autoencoder

- An autoencoder consists of both an encoder and a decoder:
  - Encoder: transform input x into latent representation z
  - Decoder: generate recovered input $\hat{x}$ from z



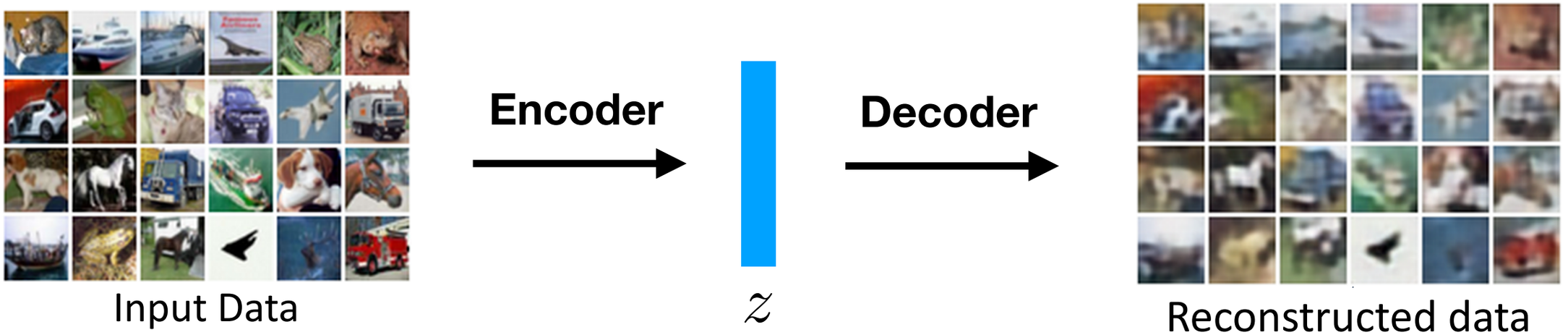
Autoencoder

- An autoencoder consists of both an encoder and a decoder:
  - Encoder: transform input x into latent representation z
  - Decoder: generate recovered input $\hat{x}$ from z



The targets are two-fold:
learn good encoder to compress the information
learn good decoder to recover the information

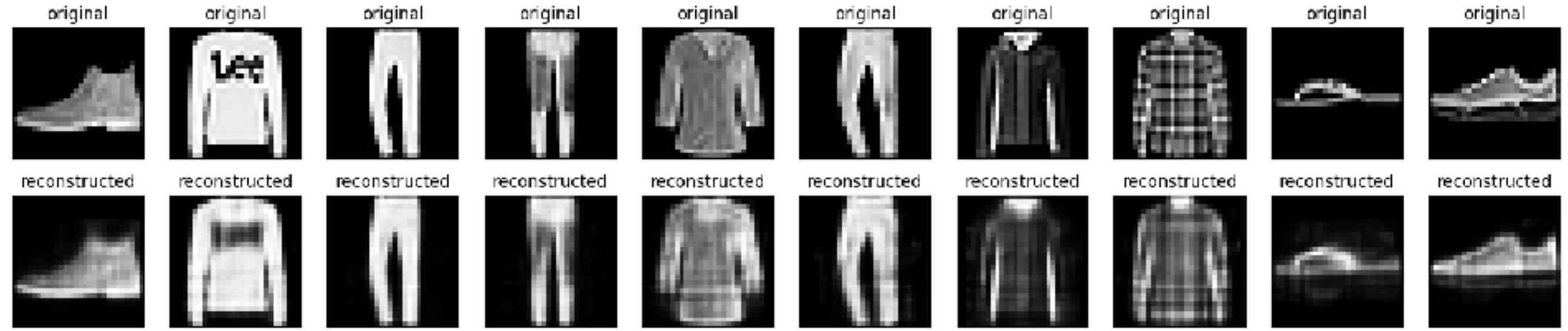
Vanilla Autoencoder



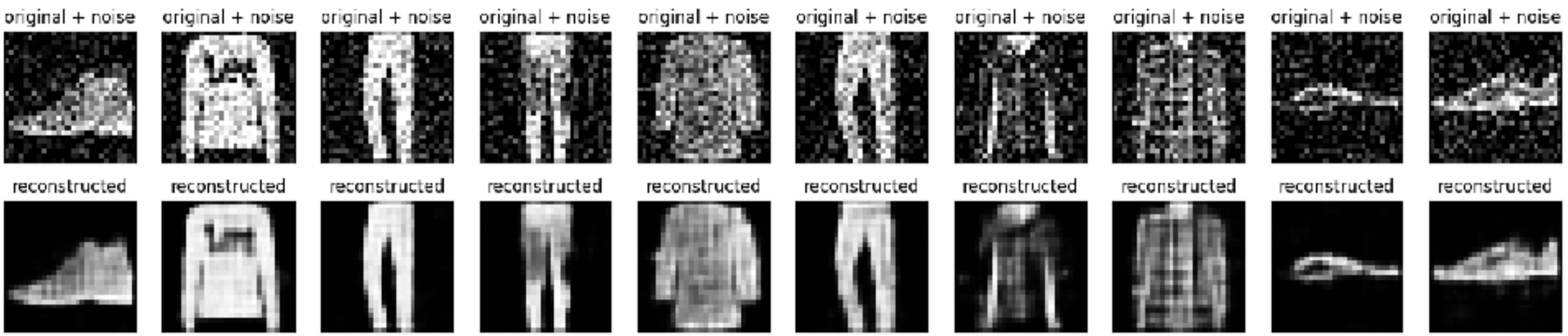
- Use NNs (Conv, MLP) to model encoder and decoder.
- Key: the dimension of z should be small for compressing information: ensure to learn useful information.
- Train with MSE loss (input-output gap) : $\|\hat{x} - x\|_2^2$

Learned Result

Recover from original

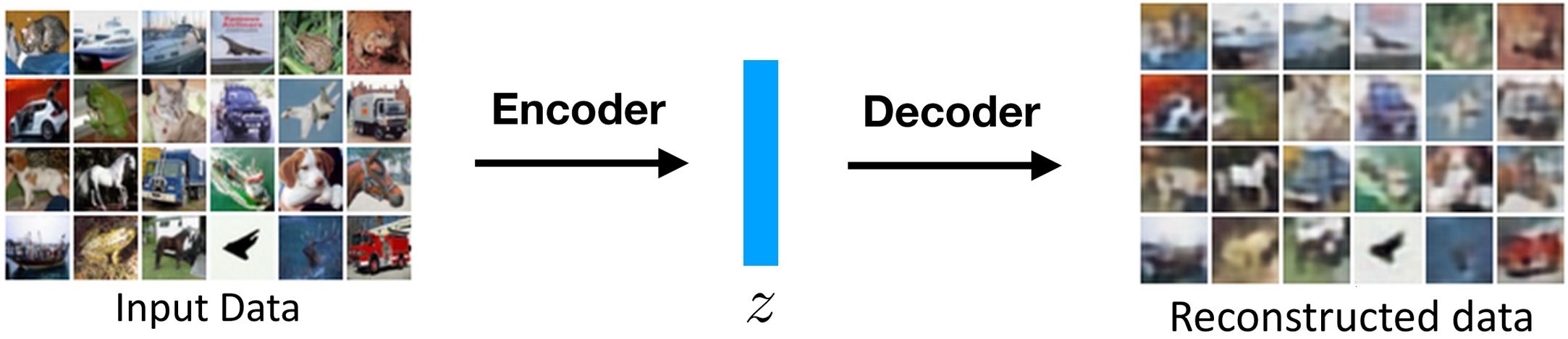


Recover from noisy (should also train with noisy data)



https://www.tensorflow.org/tutorials/generative/autoencoder

Variational Autoencoder



species,
color,
background,
weather…

Why we need a model to recover the input?
Usually, we focus on learning a good encoder:
obtain good representation of data.
Vanilla AEs are not enough. We need better modeling of the generation process.

Variational Autoencoder

Probabilistic spin on autoencoders:

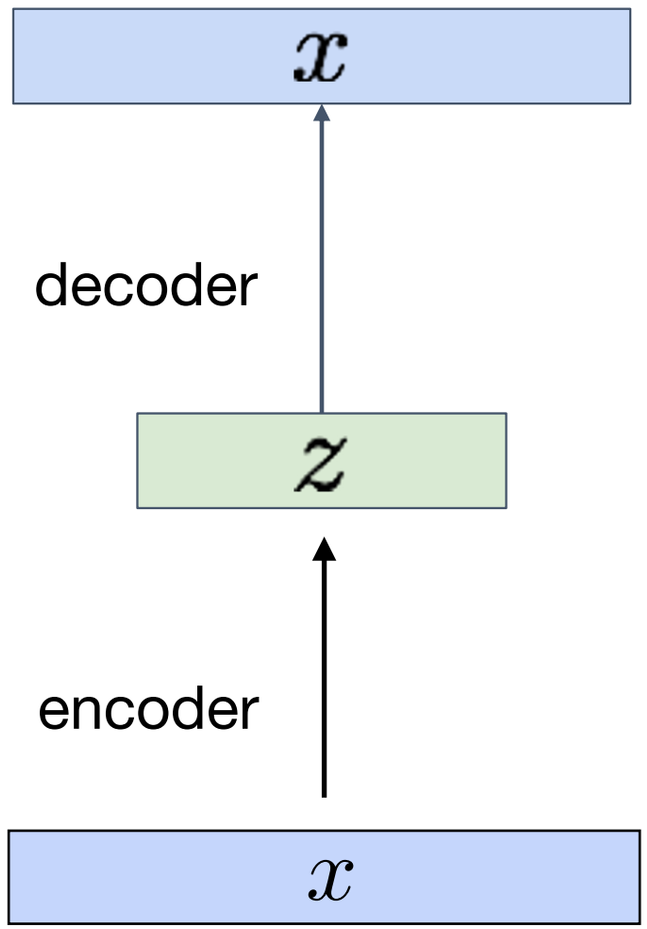
1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:

Sample from conditional $p_{\theta^*}(x \mid z^{(i)})$

Sample z from prior $p_{\theta^*}(z)$

encoder: estimate $p(z|x)$



Variational Autoencoder

Probabilistic spin on autoencoders:

1. Learn latent features z from raw data
2. Sample from the model to generate new data

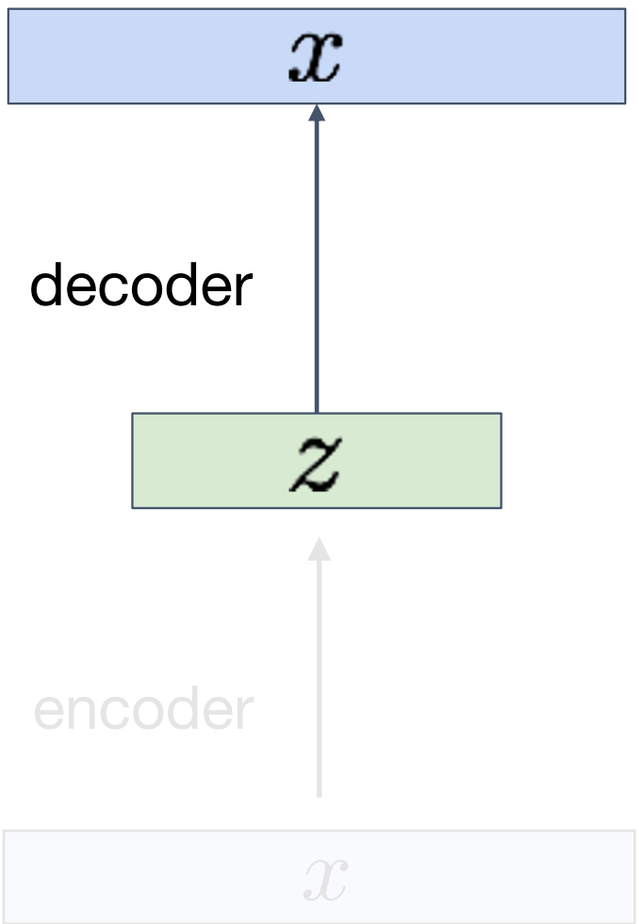
After training, sample new data like this:

Sample from conditional

$p_{\theta^*}(x \mid z^{(i)})$

Sample z from prior

$p_{\theta^*}(z)$



encoder: estimate

$p(z|x)$

Variational Autoencoder

Probabilistic spin on autoencoders:

1. Learn latent features z from raw data
2. Sample from the model to generate new data

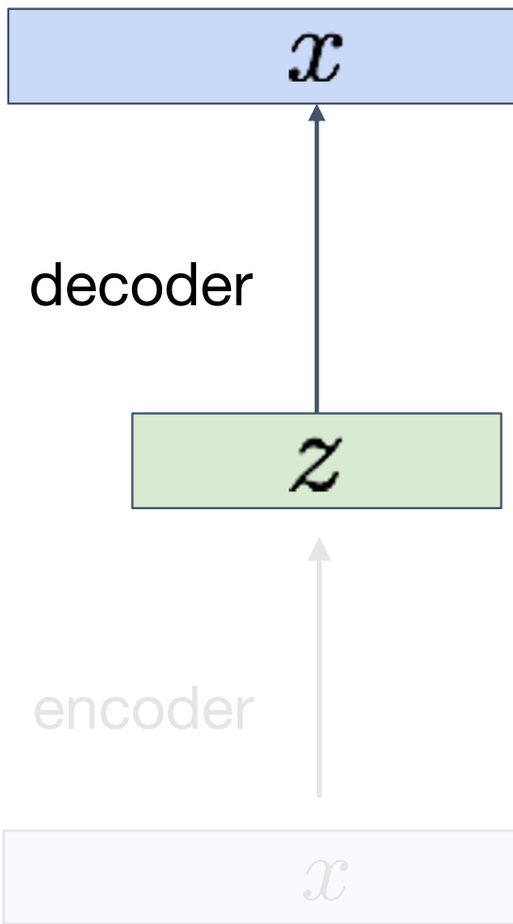
After training, sample new data like this:

Sample from conditional

$p_{\theta^*}(x \mid z^{(i)})$

Sample z from prior

$p_{\theta^*}(z)$



encoder: estimate $p(z|x)$

Assume simple prior $p(z)$, e.g. Gaussian.
Model encoder and decoder as NNs.

Variational Autoencoder

Probabilistic spin on autoencoders:

1. Learn latent features z from raw data
2. Sample from the model to generate new data

Objective: to maximize

$$E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

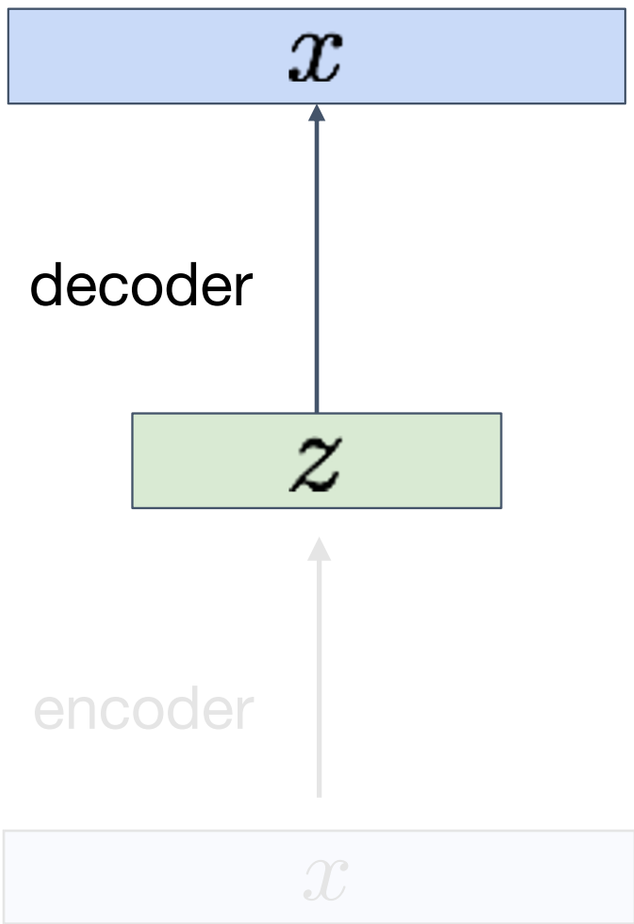
After training, sample new data like this:

Sample from conditional

$p_{\theta^*}(x \mid z^{(i)})$

Sample z from prior

$p_{\theta^*}(z)$



encoder: estimate $p(z|x)$

Assume simple prior $p(z)$, e.g. Gaussian.
Model encoder and decoder as NNs.

Variational Autoencoder

Probabilistic spin on autoencoders:

1. Learn latent features z from raw data
2. Sample from the model to generate new data

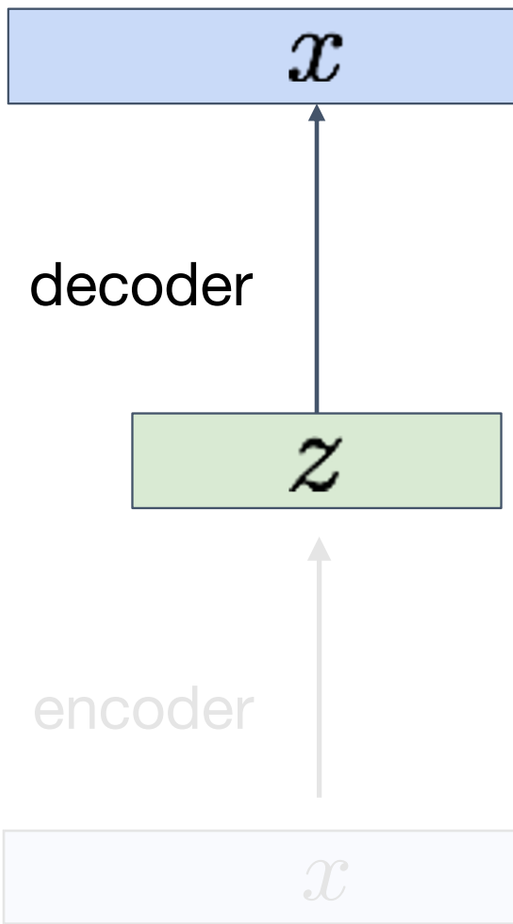
After training, sample new data like this:

Sample from conditional

$p_{\theta^*}(x \mid z^{(i)})$

Sample z from prior

$p_{\theta^*}(z)$



encoder: estimate $p(z|x)$

Objective: to maximize

$$E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

likelihood

difference between posterior and prior

Assume simple prior $p(z)$, e.g. Gaussian.
Model encoder and decoder as NNs.

Generation Results

32x32 CIFAR-10

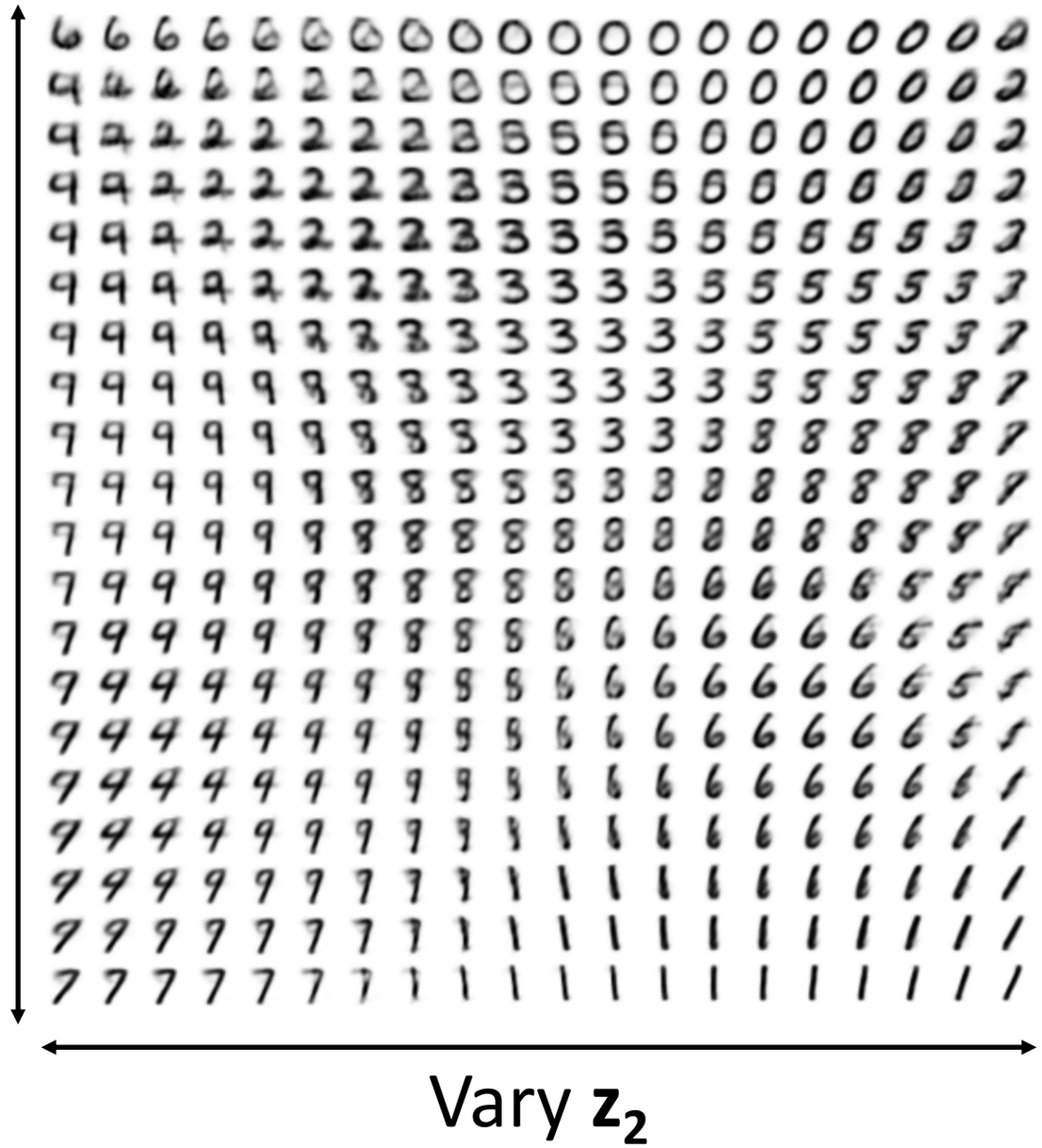
Labeled Faces in the Wild

Figures from (L) Dirk Kingma et al. 2016; (R) Anders Larsen et al. 2017.

Generation Results

The diagonal prior on p(z) causes dimensions of z to be independent

"Disentangling factors of variation"



Kingma and Welling, Auto-Encoding Variational Beyes, ICLR 2014

Latent Space Editing

The diagonal prior on p(z) causes dimensions of z to be independent

"Disentangling factors of variation"

Kingma and Welling, Auto-Encoding Variational Beyes, ICLR 2014

Latent Space Editing

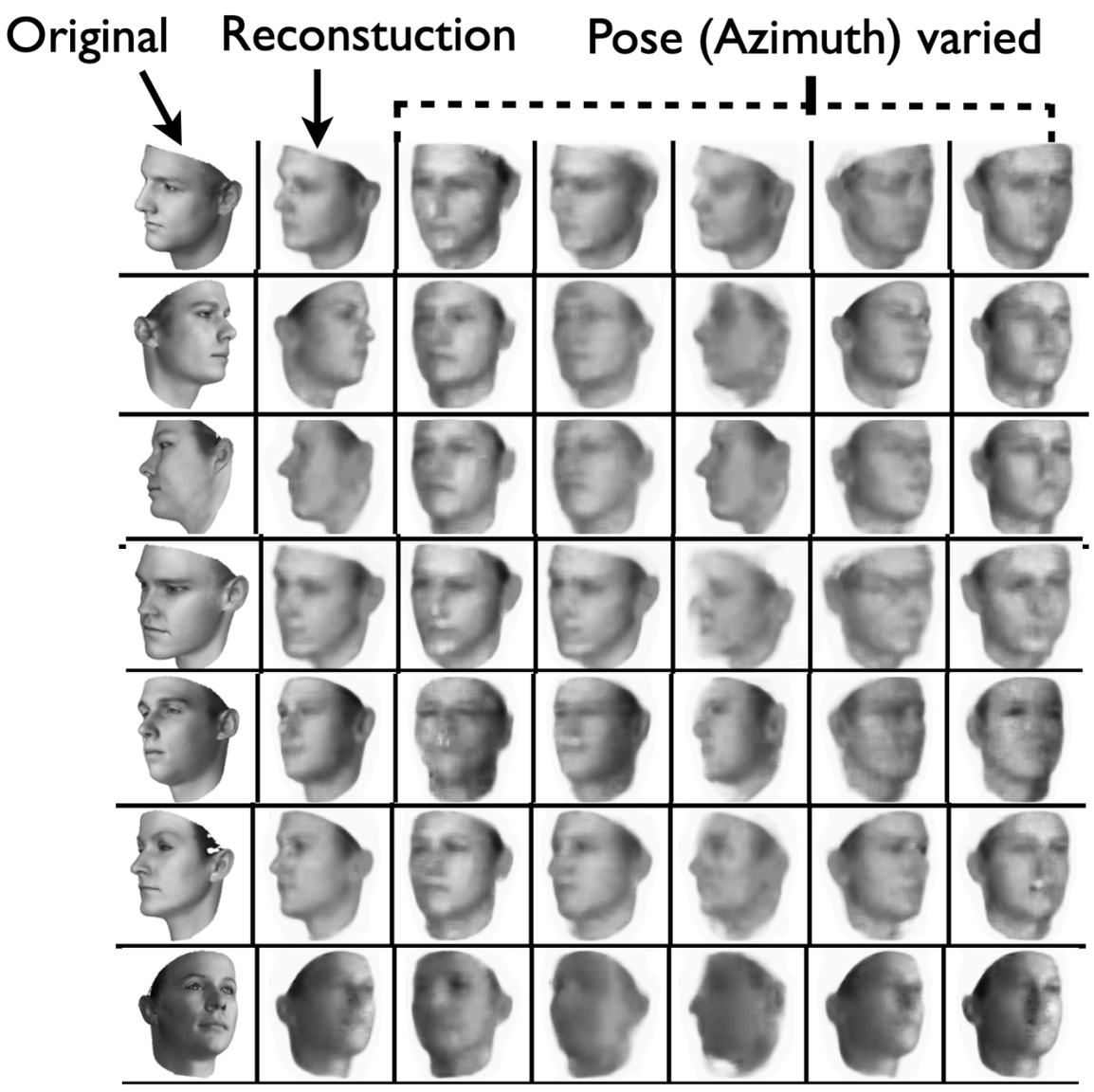
The diagonal prior on p(z) causes dimensions of z to be independent

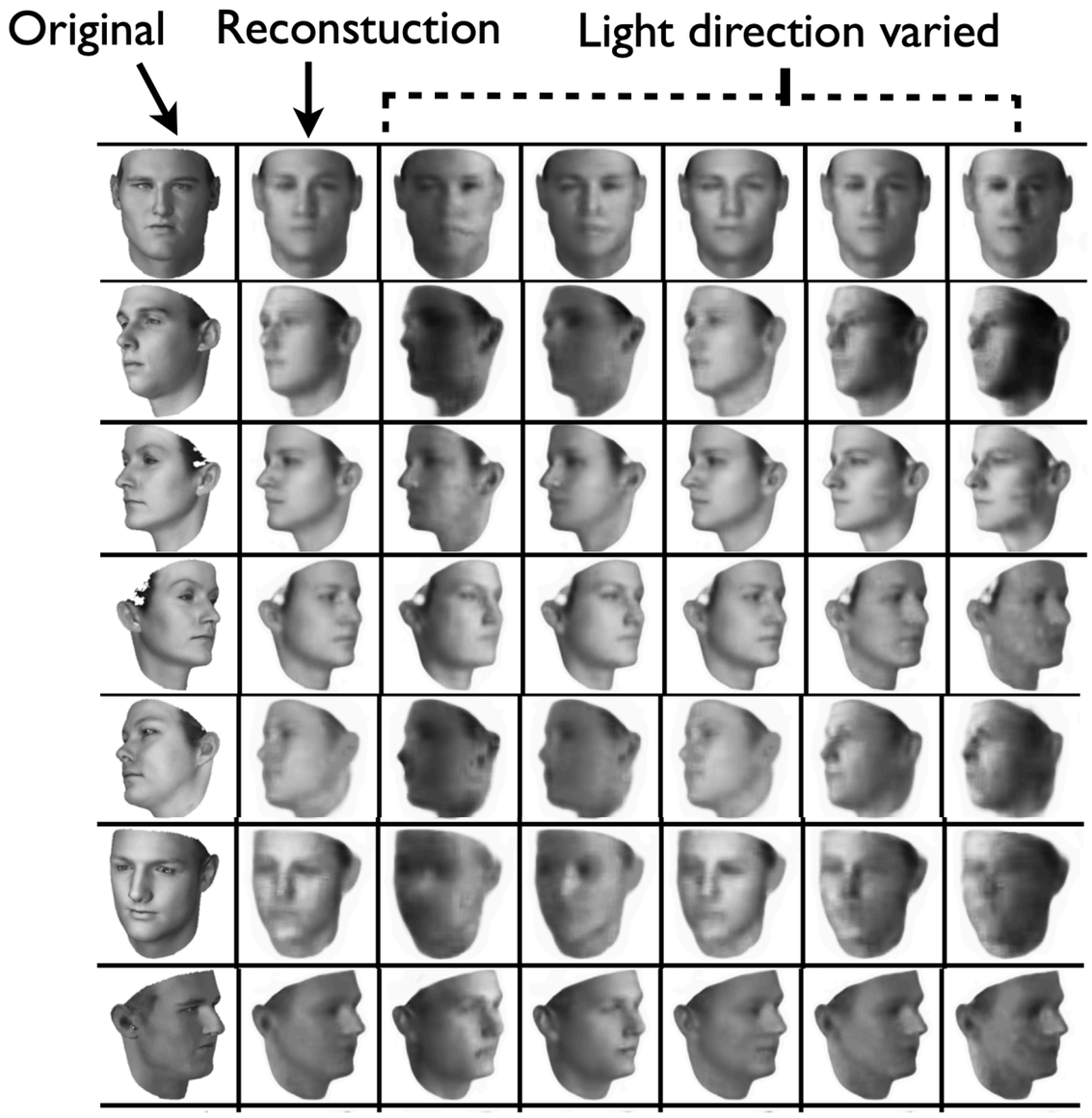
"Disentangling factors of variation"

To make the learned representations have semantic meanings, disentanglement is important.

Kingma and Welling, Auto-Encoding Variational Beyes, ICLR 2014

Latent Space Editing





Kulkarni et al, “Deep Convolutional Inverse Graphics Networks”, NeurIPS 2014